UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Detección de anomalías en series temporales de datos aeronáuticos

Samuel Valenzuela Pérez

Tutor: Doroteo Torre Toledano

Abril 2019

Detección de anomalías en series temporales de datos aeronáuticos

AUTOR: Samuel Valenzuela Pérez

TUTOR: Doroteo Torre Toledano

Grupo AUDIAS

Dpto. Tecnología Electrónica y de las Comunicaciones

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Abril de 2019

Resumen

Este Trabajo de Fin de Grado se centra en la detección de anomalías en datos de series temporales obtenidas a partir de sensores aeronáuticos. En particular, este proyecto se centra en la participación en un reto propuesto por la compañía Airbus. Las aeronaves de Airbus generan enormes cantidades de datos que generan mucha información sobre el estado de sus motores y componentes, los cuales son monitorizados por sensores durante testeos y operaciones. Al ser sometidos a cierto análisis, estos datos tienen el objetivo de ayudar a prevenir errores y facilitar el mantenimiento preventivo de las aeronaves.

Airbus propone varios retos que consisten en clasificar series temporales como anomalías o normales. En concreto, todo este trabajo se centra en uno de los retos: detección de anomalías en datos de acelerómetros de helicópteros Airbus. Se proporciona un conjunto de datos con secuencias temporales que se consideran normales, a partir del cual se propone entrenar un modelo que aprenda a analizar diferencias con otras secuencias y a clasificarlas como anomalías o no. La detección de anomalías con antelación a la ocurrencia de fallos en el sistema es de gran utilidad especialmente tratándose de aeronaves de gran valor que requieren alta fiabilidad.

El reto principal del trabajo es el de no poder entrenar modelos de manera supervisada ya que se carece de un dataset de entrenamiento con secuencias anómalas. Esto nos encauza a investigar técnicas para la detección de anomalías distintas de modelos de aprendizaje automático supervisado. Se deberá analizar profundamente el dataset proporcionado, y clasificado como datos normales por expertos de Airbus, para concluir las mejores maneras de procesar los datos y medir el grado de anomalía de cada secuencia temporal. Una vez desarrollado un modelo, se evaluará y se corregirá en base a los resultados de la evaluación con otro conjunto de datos proporcionado en el concurso, cuya cantidad de secuencias anómalas y normales es desconocida a priori.

Palabras clave

Anomalías, series temporales, Airbus, aeronave, acelerómetro, aprendizaje automático, inteligencia artificial, modelo, supervisado, no supervisado, clasificación, señales.

Abstract

This Bachelor's thesis concentrates on anomaly detection on time series datasets obtained from aircraft's sensors. In particular, this project is based on participating on a challenge proposed by the aircraft company Airbus. Their aircrafts generate massive amounts of data providing great information about the state of its engines and components, which are monitored during tests and operations. This data is put under an analysis with the goal of helping to prevent the occurrences of errors and to facilitate the aircraft's maintenance in advance to them.

Airbus proposes several challenges which consists on anomaly detection in time series data. In specific, the work is concentrated around one of the challenges: Airbus Helicopters Accelerometer challenge. A dataset is given with sequences that are considered normal with which a model should be trained to learn to differentiate between normal and anomaly sequences. Anomaly detection prior to system failures has high utility especially when it comes to complex aircrafts that require great reliability.

The greates challenge of the task is to work on an unsupervised context. It is unsupervised because the given dataset contains only what are considered normal sequences by Airbus

experts. Working with a dataset that does't contain all of the classes, the key will be to analyze it to infer techniques to calculate the degree of anomaly of each time sequence. There is also provided a validation dataset, and once each model is developed, they will be evaluated and corrected based on the results of classifying the time sequences of this dataset, whose number of anomalies is, a priori, unknown.

Keywords

Anomaly, time series, Airbus, aircraft, accelerometer, machine learning, artificial intelligence, model, supervised, unsupervised, classification, signals.

Agradecimientos

Gracias a Doroteo por su colaboración y por guiarme a lo largo del proyecto.

Gracias a Guillermo por su ayuda en varios puntos.

Gracias a Elias y a Freddy por sus revisiones y consejos.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

Este trabajo de final de grado consiste en la participación en un reto tecnológico internacional de la compañía Airbus en el que cualquier persona, grupo de investigación, empresa o universidad puede participar, con el fin de competir con expertos de todo el mundo en el tema propuesto. Se participa en el reto representando al grupo de investigación AUDIAS, el cual se centra en tratar este tipo de problemas de la inteligencia artificial, como es el análisis de series temporales, lo cual es una motivación por trabajar de manera consistente para conseguir desarrollar una buena solución al problema planteado. La implicación del grupo de investigación en retos con características similares y la experiencia del tutor del proyecto en el área permiten una buena base de donde partir.

Airbus propone este reto que, por un lado, busca la aplicación de la inteligencia artificial para entender mejor los datos recogidos de sus aeronaves, tratarlos y conseguir una o varias buenas soluciones para detectar anomalías en el comportamiento de los sistemas. Por otro lado, Airbus también persigue dar reconocimiento a los mayores expertos actuales en el mundo de análisis de series temporales, y también fomentar a la comunidad a implicarse en la investigación para resolver problemas específicos relacionados con la industria aeroespacial.

Personalmente, parte de la motivación viene de un entusiasmo por conocer más sobre en mundo del análisis de datos, aprendizaje automático, limpieza de datos, tratamiento de señales temporales y otros temas relacionados. Toda esta área de la informática es suficientemente interesante y profunda como para atraer a ingenieros a dedicarle toda una carrera, y esto me anima a investigar más sobre esta parte de la ciencia de la computación, dándome la posibilidad de dedicarme a ello en el futuro. El tema de la inteligencia artificial y en especial los modelos de aprendizaje automático, son estudiados con profundidad durante la carrera, lo que también me incita a seguir aprendiendo sobre modelos más sofisticados y sobre todo el proceso que supone un estudio completo de un problema como el propuesto en este concurso.

El mayor reto del trabajo es el de no poder entrenar modelos de manera supervisada, ya que se carece de un conjunto de datos con individuos de todas las clases a clasificar. Este reto consiste en resolver el problema de clasificación de los datos como anómalos o normales sin tener un conjunto de datos de entrenamiento que contenga ambas clases. El conjunto de entrenamiento proporcionado solo contiene individuos considerados normales (no anómalos) por expertos de Airbus.

1.2 Objetivos

Los objetivos de este trabajo de fin de grado son los siguientes:

1. Comprender el problema propuesto en el concurso, sus reglas y el método de validación de las soluciones.
2. Generar un marco de acción adecuado en torno a las fechas en las que se hacen públicos los datos de entrenamiento y las fechas de entrega de soluciones.
3. Realizar un análisis profundo de los datos que forme una base para investigar y proponer los modelos más adecuados.

4. Estudiar el estado actual del arte sobre los modelos usados en la comunidad para el problema a resolver, así como la familiarización con las herramientas más utilizadas para la implementación de modelos de aprendizaje automático.
5. Utilizar los distintos modelos que se consideren adecuados para resolver el problema al que nos enfrentamos con el objetivo de comparar sus resultados y entender sus diferencias y ventajas.
6. Proponer, finalmente, dos modelos que resuelvan el reto del concurso notablemente.
7. En general, se busca una aprender a desarrollar un proceso de trabajo para enfrentar cualquier tipo de reto relacionado con la inteligencia artificial, así como adquirir las habilidades necesarias para ser un participante competente.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- *Estado del arte:* consiste en una revisión de la literatura académica y documentos de investigación publicados en la actualidad sobre el tema que rodea el problema a resolver. Se analizan técnicas de modelado de datos y aprendizaje automático que son utilizadas hoy en día y que serán estudiadas durante el proyecto.
- *Diseño:* se describe con detalle el concurso y, por otro lado, las herramientas utilizadas durante el desarrollo del trabajo.
- *Desarrollo:* se describe cronológicamente todas las tareas realizadas, así como los razonamientos acerca de los modelos utilizados, estudio de los datos y sus resultados.
- *Integración, pruebas y resultados:* se describen todas las pruebas realizadas a lo largo del desarrollo y sus correspondientes resultados con el fin de realizar una comparación para conseguir la solución óptima.
- *Conclusiones y trabajo futuro:* se resume finalmente el resultado del proyecto, sus consecuencias, y se describen caminos a seguir para continuar investigando y experimentando sobre el tema.

2 Estado del arte

2.1 Modelado de señales temporales

2.1.1 Modelado estadístico

El primer paso para entender los datos y su forma es la realización de un modelado estadístico o formalización matemática del comportamiento de las variables y sus relaciones entre ellas. Se realiza un estudio, normalmente sobre una muestra aleatoria del dataset que represente la forma general de los datos, y a partir de este estudio, se realizarán predicciones -usando la probabilidad- del resto de las muestras del dataset, asumiendo una distribución o relación entre los datos.

La media aritmética de una muestra se realiza para observar el promedio de los valores o su distribución.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Por otro lado, la desviación estándar nos sirve para medir la variación o dispersión de los datos. Si tiene un valor cercano a cero significa que la dispersión es baja y los datos son muy cercanos a su media aritmética. En cambio, mientras más grande sea el valor, más dispersión habrá. Consiste en la raíz cuadrada de la media de los cuadrados de las desviaciones desde la media.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})}{n}}$$

2.1.1.1 Coeficiente de correlación

Para hallar la relación entre dos variables se utiliza el coeficiente de correlación. El coeficiente de correlación de Pearson mide linealmente el grado de dependencia entre variables de manera independiente de la escala y la dispersión de cada una.

$$\rho_{X,Y} = \frac{\sigma_{XY}}{\sigma_X\,\sigma_Y} = \frac{\mathrm{E}[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X\,\sigma_Y}$$

El valor resultante es lineal entre [-1, 1] siendo 1 la mayor correlación posible, es decir, que dependen completamente una variable de otra, y -1 cuando no dependen en absoluto.

2.1.1.2 Modelo autorregresivo

En el estado del arte del modelado de series temporales de datos se usan los modelos autorregresivos de media móvil o ARMA. Este modelo sirve para entender el comportamiento de la serie temporal y a predecir futuros valores. Consta de dos partes: modelo autorregresivo (AR) y modelo de media móvil (MA).

El **modelo autorregresivo** AR(p) busca modelar la serie temporal como función de sus valores anteriores. Consiste en un filtro de respuesta infinita al impulso (IIR) donde si la señal de entrada es una señal de impulso, la salida del filtro es un número infinito de términos no nulos. AR(p) da el valor de la serie en el momento *t* como combinación lineal de los

últimos *p* registros de la variable. En la definición del modelo, $\vec{\phi}$ es un parámetro, c una constante y ϵ_t un término de error de ruido blanco.

$$X_t = c + \sum_{i=1}^{P} \phi_i X_{t-i} + \epsilon_t$$

El **modelo de media móvil** MA(q) consiste en un filtro de respuesta finita al impulso (FIR). En un tiempo *t,* el valor de la serie se expresa como una combinación de innovaciones. Este modelo, al final, especifica que la salida de la variable depende linealmene del valor actual y anteriores. En la definición del modelo $\vec{\theta}$ es un parámetro y $\vec{\varepsilon}$ errores de ruido blanco.

$$X_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q}$$

EL modelo ARMA se aplica con datos estacionarios cuyas propiedades estadísticas, como la media y variancia, son constantes a lo largo del tiempo. Por otro lado, existe tambien el modelado ARIMA aplicado a series temporales no estacionarias. Funciona de la misma manera reemplazando la serie de datos con transformaciones matemáticas para hacerla estacionaria [1].

2.1.1.3 Modelo oculto de Markov

En el modelado de series temporales, los modelos de Markov encajan muy bien para entender y predecir el comportamiento de los datos y sus cambios a lo largo del tiempo. El modelo se basa en asumir que existe un número finito de estados entre los cuales hay transiciones. Solo uno es el estado actual de la secuencia observada en un tiempo *t* y desde el cual pueden o no ocurrir transiciones a otros estados. Se asume, por otro lado, que los estados futuros dependen solo del estado actual y no de los estados anteriores [2]. Se puede definir la probabilidad de ir a un estado en un tiempo *t* dependiendo del estado en el tiempo *t -1* como $p(y_t|y_{t-1})$.

Uno de los modelos de Markov es el modelo oculto de Markov. En el modelo de Markov los estados son directamente visibles para el observador, pero en el modelo oculto de Markov los estados no son visibles, sino que se obtienen unas salidas u observaciones que dan información sobre los estados. A partir de estas observaciones, el objetivo es el de predecir el estado o, hablando de series temporales, clasificar una secuencia. Otras utilidades, por ejemplo en procesamiento de voz, es la de predecir la siguiente palabra que va a decir un locutor a partir de la anterior. En general, a través de unas observaciones se intenta captar la naturaleza oculta de una serie temporal que consiste, básicamente, en la serie de estados. Se puede representar la relación entre una observación x_t y el estado y_t como $p(x_t|y_t)$.

Para realizar el entrenamiendo del modelo oculto de Markov se utiliza el algoritmo Foward-Backward. En el primer paso del algoritmo, es decir, en Foward, se calcula el conjunto de probabilidades de acabar en cualquier estado dadas las primeras *t* observaciones de la secuencia. En el segundo paso, Backward, se calcula el conjunto de probabilidades de observar el resto de observaciones dado cualquier punto de inicio *t*. Estos dos conjuntos de probabilidades se combinan para obtener la distribucion de los estados en un tiempo determinado dada la observación de la secuencia. Los parámetros del entrenamiendo son: tamaño de la ventana deslizante sobre las observaciones (cuántas observaciones envuelve la condición markoviana), el número de estados que existen y el número de iteraciones del algoritmo. El objetivo es la clasificación de una secuencia completa, la predicción del estado. El modelo HMM (por sus siglas en inglés, *Hidden Markov Layers*) es muy utilizado en la comunidad para aplicarlo a series temporales con el objetivo de realizar predicciones con entrenamiento supervisado y no supervisado.

2.1.2 Modelado espectral

Un fenómeno ondulatorio, como el sonido, con características como la frecuencia y la amplitud, se compone de una onda fundamental o primer armónico que es la frecuencia con mayor amplitud y otro conjunto de armónicos cuya amplitud va decreciendo. Los armónicos son ondas sinusoidales cuyas frecuencias son múltiplos enteros positivos de la frecuencia original de la onda fundamental. Para realizar un análisis de una señal, se descompone en sus partes más simples, en sus armónicos y onda fundamental. Esta descomposición se denomina espectro.

Para realizar un análisis espectral, se representa la señal como la transformada de Fourier de amplitud $A = A(v)$. Básicamente la señal se representa como una función de su amplitud dependiendo del tiempo.

$$s(t) = \int_{\mathbb{R}} A(v) e^{-2\pi i v t} d\omega$$

La transformada de Fourier permite realizar la descomposición espectral de la onda en sus armónicos.

2.1.2.1 Espectrograma

El espectrograma es una herramienta muy usada para analizar señales mediante sus espectros y representarlas con respecto al tiempo, la frecuencia y la amplitud. Se usa en áreas como música, procesamiento de voz, sismología, sonar y radar.

Consiste en una representación de la señal en tres dimensiones. En el eje X se representa el tiempo, y en el eje Y se representan las frecuencias. Por último, las áreas con mayor intensidad de gris representan una amplitud mayor y de menor intensidad, menor amplitud. Para calcularlo, se utiliza una ventana deslizante que determina un número específico de secuencias, de las cuales se calcula la transformada de Fourier para conseguir así su espectro.

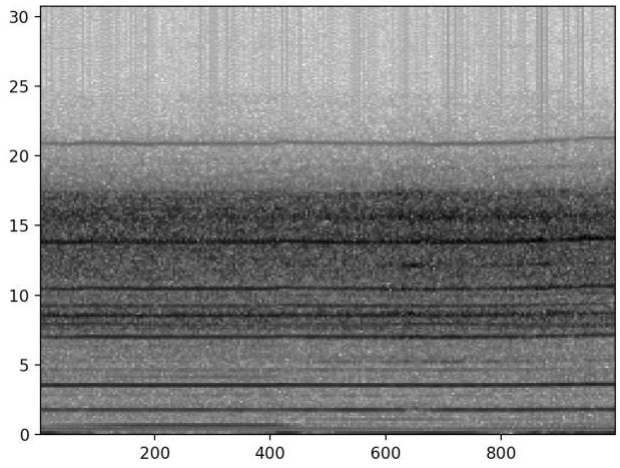

Figura 2-1: espectrograma

En la figura 2-1 se observa un ejemplo de un espectrograma mostrando las amplitudes con la intensidad en una escala de grises. Se puede observar una amplitud predominante en la frecuencia de aproximadamente 2 Hz. El espectrograma se representa como una matriz con valores de escala de grises. Esta matriz se puede tratar como una imagen que se puede modelar con técnicas de aprendizaje automático del estado del arte como son las redes neuronales convolucionales, de lo cual se habla en la sección 2.2.2.1.

2.2 Modelos de aprendizaje automático

El modelado de datos con algoritmos de aprendizaje automático, como subcampo de la inteligencia artificial, se describe como la realización de tareas sin ser especificadas con instrucciones explícitas, sino basadas en el reconocimiento de patrones. La clasificación entre dos clases, o también denominada clasificación binaria o bipolar, consiste en separar los datos en dos categorías en base a una serie de reglas que toman en cuenta las características de cada individuo. La clasificación se da en una gran variedad de problemas como testeos médicos, clasificación de correos electrónicos como spam o no spam o detección de anomalías. A continuación se describen algunos algoritmos que encajan con este tipo de clasificación y con el problema a resolver en el proyecto.

2.2.1 Modelado con K vecinos próximos

El algoritmo de K vecinos próximos es muy utilizado en la comunidad para una gran variedad de tareas. A pesar de que es más conocido por clasificación supervisada como algoritmo de regresión.

Dado un conjunto de parámetros como representación de un ejemplo a clasificar, K vecinos próximos clasifica este ejemplo comparándolo con todos los ejemplos con los que se ha entrenado el modelo. Se clasifica el nuevo individuo con la clase predominante en los K vecinos más próximos. Para determinar la cercanía entre dos ejemplos se calcula la distancia entre ellos, por ejemplo, la distancia euclídea, la cual es una de las más utilizadas. Siendo ejemplo representado con un vector $\vec{x} = (x_1, x_1, \ldots, x_p)$ la distancia euclídea se calcula de la siguiente manera.

$$d(x, y) = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}$$

Para realizar la regresión en lugar de la clasificación, el método más común es el de realizar una media de las distancias a los K vecinos más próximos, aunque también se pueden utilizar otros cálculos como el sumatorio de todas las distancias.

2.2.2 Modelado con Redes Neuronales Artificiales

Uno de los modelos más utilizados hoy en día para el aprendizaje automático son las redes neuronales artificiales. Se inspiraron inicialmente en los procesos básicos de las redes neuronales biológicas como el procesamiento distribuido en neuronas y el aprendizaje a través de las conexiones entre ellas. Estas conexiones entre neuronas transmiten datos que cada neurona recoge de sus enlaces entrantes y a los cuales aplica una función de activación para determinar su señal de salida a las siguientes neuronas.

Las redes neuronales artificiales tienen una gran variedad de aplicaciones en la ciencia de datos como la clasificación, almacenamiento y recuperación de datos, clusterización, optimización y predicción, entre otros. En este apartado se detallan las características y el funcionamiento de las redes más usadas en el estado del arte y que son de importancia para este proyecto.

2.2.2.1 Redes neuronales convolucionales

Los avances en el modelado de imágenes con el aprendizaje profundo ha sido construido y perfeccionado a lo largo del tiempo, aunque siempre alrededor de un algoritmo en particular, las redes neuronales convolucionales (CNN). Las redes CNN son capaces de aprender a

diferenciar imágenes mediante el reconocimiento de patrones simples y complejos. La preferencia de redes convolucionales sobre otro tipo de redes se da por la capacidad de aprender teniendo en cuenta las relaciones de los píxeles con sus píxeles vecinos y por la reducción de parámetros y reutilización de pesos, lo cual es de gran importancia cuando las imágenes son grandes.

En una capa convolucional las neuronas se representan por filtros. Cada filtro consiste en una matriz pequeña con un patrón, como puede ser una línea recta vertical, el cuál se busca en la imagen. Cada neurona utiliza su filtro como una ventana deslizante sobre la matriz de entrada. En cada una de las posiciones multiplica elemento por elemento los valores de cada matriz y finalmente los suma, obteniendo un valor que representa la similitud entre ambas matrices. Este valor se guarda en la salida, que junto con los demás valores que la ventana deslizante va generando, forma una matriz de salida.

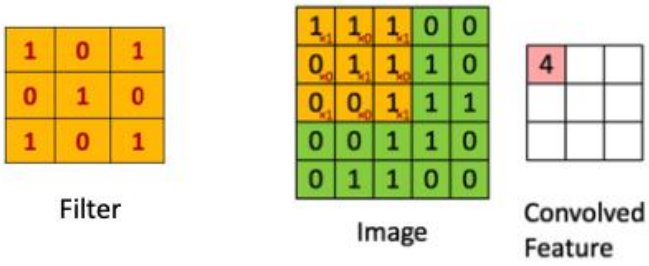

Figura 2-2: Funcionamiento de un filtro

La figura 2-2 muestra el cálculo de la primera ventana del filtro, donde coinciden cuatro píxeles y como cada píxel tiene valor uno, el resultado es cuatro. Cada capa convolucional consta de varios parámetros a partir de los cuales se puede calcular el tamaño de la salida con la siguiente fórmula [3].

$$(W - F + 2P)/S + 1$$

El valor W representa el tamaño de la entrada. La F representa el tamaño de los filtros. El valor S representa el número de píxeles que avanza la ventana en cada paso. Alrededor de la imagen se puede colocar un margen de ceros, cuyo tamaño se representa con P. Se puede calcular de esta manera la salida del ejemplo anterior como:

$$(5x5 - 3x3 + 0)/1x1 + 1 = 3x3$$

Las capas pooling son otro elemento imprescindible en las CNN. Al tener una gran cantidad de parámetros de entrada, estas redes necesitan reducirlos a lo largo de la red tanto para reducir los parámetros que se entrenan como para discriminar los datos importantes en el cálculo de la salida de la red. Para esto sirven las capas pooling, que al igual que las capas convolucionales, parten de una ventana deslizante de la cual sale un valor. Este valor se escoge dependiendo del tipo de capa:

- Max pooling: se elige el valor máximo entre los valores que comprende la ventana en la imagen. Es el más utilizado [4].
- Min pooling: se elige el valor mínimo que comprende la ventana.
- Average pooling: se elige la media de los valores que comprende la ventana.

En caso de que la ventana deslizante tenga un tamaño de 2x2 y la S sea 2x2, la imagen tendrá una reducción en un 75% de su tamaño. La capacidad de reducción causa pérdida de

información, pero ésta se minimiza porque la manera de calcular la salida mantiene las características importantes de la imagen sin cambiar mucho sus posiciones dentro de esta.

Las redes convolucionales, como se ha explicado, pueden aprender a localizar líneas rectas o patrones sencillos en la primera capa convolucional y los patrones son más complejos a medida de que se van añadiendo más capas. Este funcionamiento es similar al del cortex visual de un ser humano.

2.2.2.2 Redes neuronales residuales

Son redes más profunas, lo cual tiene ciertas ventajas ya que se consigue más generalización. El entrenamiento en redes muy profundas se complica notablemente. Un ejemplo de esto es un problema conocido por el nombre de desvanecimiento del gradiente, que consiste en una parada del entrenamiento de la red porque los ajustes de los pesos de las conexiones entre neuronas son prácticamente nulos debido a un gradiente de la función de error demasiado pequeño. Las redes ResNet lo resuelven mediante conexiones nuevas o atajos entre capas. La idea es que en cada bloque de varias capas haya dos caminos para los datos. El camino principal es el de las capas de la red sin ningún cambio y el segundo camino es un atajo entre la entrada del bloque y su salida. Antes de la función activación del bloque, se suman los resultados de ambos caminos ajustando los tamaños de la matriz del segundo camino para que la suma sea válida.

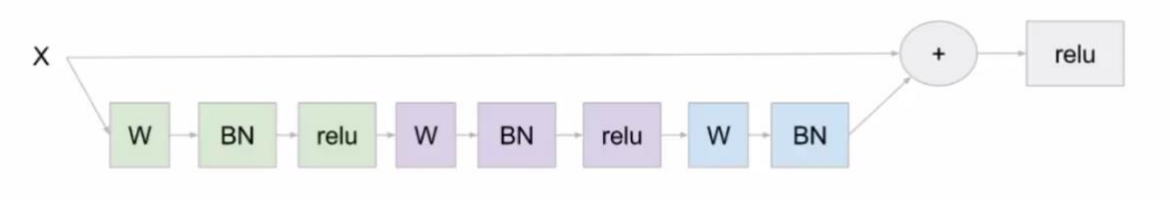

Figura 2-3: Bloque ResNet típico

En conclusión, las redes neuronales residuales son una mejora que permite construir redes mucho más profundas mejorando el acierto del modelo y, aunque se basa en las redes convolucionales, es aplicable a otros tipos de redes. Sin embargo, se ha probado que estas redes han sufrido reducciones de acierto al añadir más de 1000 capas, surgiendo problemas como el del desvanecimiento del gradiente. Una de las técnicas usadas es la siguiente: Se elimina la función de activación final permitiendo que el siguiente bloque de capas reciba directamente el resultado de la suma de los caminos. Por otro lado, los caminos principales de los bloques comienzan con una normalización y una función de activación antes de las capas convolucionales. Esto permite que la entrada de la red pueda llegar intacto a cualquiera de los bloques ResNet y se resuelve por completo el desvanecimiento del gradiente. Con este último cambio se consiguen mejoras en el acierto con redes de más de 1000 capas.

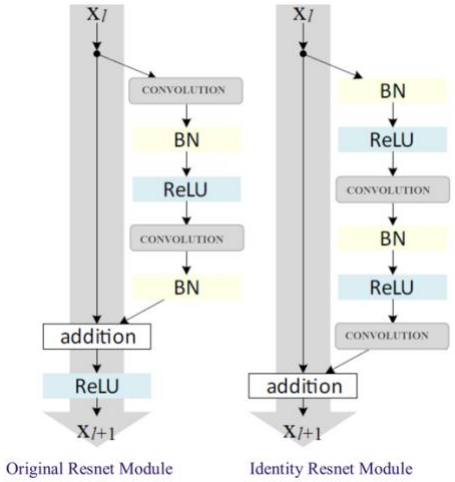

Figura 2-4: Comparación Bloques ResNet

Esta nueva arquitectura de los bloques se denomina *Identity ResNet Module* y en la figura 2-4 se compara visualmente con los bloques normales [5].

2.2.2.3 Redes neuronales residuales anchas

Las redes neuronales residuales definidas en el anterior punto muestran mejoras en los aciertos de los modelos al añadir más capas a las redes llegando a escalar hasta tener miles de capas. Sin embargo, cada pequeña mejora en el acierto se obtiene prácticamente multiplicando por dos el número de capas en la red. Esto genera una cantidad masiva de parámetros a entrenar, lo que disminuye notablemente la eficiencia del entrenamiento, a pesar de que las capas de las redes residuales se intenten hacer lo más pequeñas posibles. Estos cambios para disminuir el coste del entrenamiento termina siendo una debilidad de este tipo de redes. Además, como se describe en [6], ocurre que la información transmitida entre bloques disminuye, dando preferencia a los caminos residuales y dejando inutilizados muchos bloques de la red. Una de las técnicas usadas para la solución de este problema de describe en [7]. La idea consiste en, en lugar de aumentar la profundidad de las redes, se busca cuál es la anchura máxima idónea de las redes. Se prueba la validez de esta teoría con experimentos comparativos entre esta nueva arquitectura con otras redes residuales mucho más profundas y se obtienen notables mejoras en los resultados. Al final, se mejora el acierto del modelo con redes mucho menos profundas y con capas de gran anchura.

La arquitectura de las redes residuales anchas o WRN se compone de una capa convolucional inicial de tamaño fijo, seguida de tres bloques residuales de capas convolucionales con un factor k que multiplica el número de parámetros en las capas convolucionales. Además de realizar cambios en el factor k, se puede modificar también el número de capas convolucionales total de la red. Con una notación WRN-n-k siendo n el número de capas convolucionales, se prueba que el modelo WRN-40-4, con menos parámetros que la red ResNet de 1001 capas [8]obtiene mejores resultados que ésta en la clasificación del dataset CIFAR-100 [9].

2.2.2.4 Redes neuronales LSTM

Las redes neuronales recurrentes (RNN) son una clase de arquitectura de redes neuronales artificiales que usan funciones iterativas para guardar información en ellas. Son flexibles en el uso de los datos de entrada permitiendo ganar información del contexto y aprendiendo

qué datos almacenar y cuáles ignorar. Suelen ser usadas en la clasificación de secuencias de series temporales.

Sin embargo, las redes RNN son deficientes a la hora de guardar información por largos periodos de tiempo. Las redes *Long Short-Term Memory* (LSTM) consisten en un rediseño basado en las redes neuronales recurrentes que logran resolver el problema del desvanecimiento del gradiente a lo largo del tiempo. Son muy usadas para el modelado de señales temporales como la voz y la escritura manuscrita (considerándola como una señal temporal) [10].

3 Diseño

3.1 Entorno y reglas del concurso de Airbus

Como se ha dicho en la introducción, el proyecto está completamente enfocado en el concurso de Airbus de detección de anomalías en series temporales. En este apartado se detallan los aspectos importantes del concurso a tener en cuenta durante el desarrollo.

3.1.1 Programa

El concurso gira en torno a la plataforma creada por Airbus llamada *AirbusAI Gym platform* donde los participantes pueden realizar las inscripciones, obtener los datasets, realizar entregas de resultados de test así como participación en foros de discusión acerca del concurso. En total hay tres concursos sobre series temporales con la misma estructura incluyendo las fechas generales del programa. Como se ha mencionado antes, el proyecto se centra en uno de ellos, el cual es denominado AH en la figura 3-1 como abreviación de *Airbus Helicopter*.

El programa empieza el 7 de enero de 2019, día a partir del cual los participantes pueden empezar a inscribirse en la plataforma. Dos días más tarde, 9 de enero, se desvelan los datasets para dar comienzo a la fase de entrenamiento en la cual los participantes construyen sus modelos. Esta fase termina el 2 de mayo de 2019, y en esta misma fecha comienza la fase de entregas donde cada equipo debe entregar dos soluciones al problema hasta el día 15 de mayo. Finalmente, el 28 de junio de 2019 se realizará un taller en las oficinas centrales de Airbus en Toulouse, Francia, para reunir a los participantes y, en especial, para dar lugar a las presentaciones de los proyectos de los ganadores del concurso.

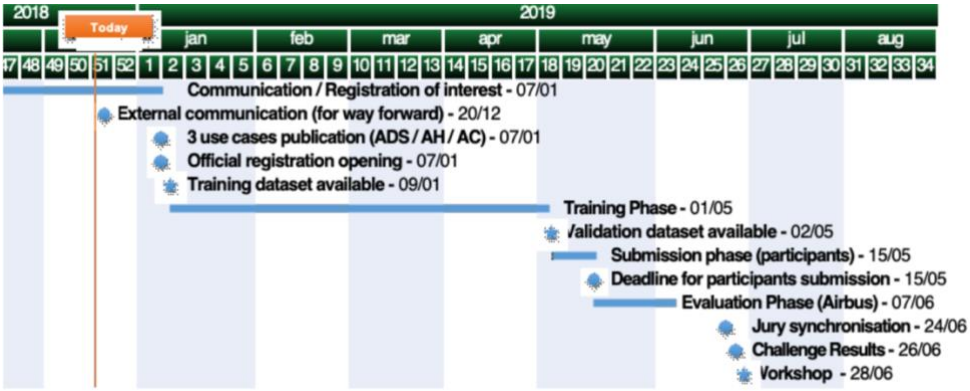

Figura 3-1: Programa del concurso

De manera especial, el concurso AH no solo proporciona el dataset de entrenamiento al inicio de la primera fase, sino también el dataset de validación. El dataset de validación es aquél para el cual se deben realizar las clasificaciónes de sus datos y entregar dichas clasificaciones como resultado del modelo del equipo participante. Sin embargo, se podrá hacer una entrega de solución de validacion de manera diaria, y no solamente a partir del 2 de mayo. Esto significa que los equipos obtendrán los resultados de sus modelos en validación, a lo sumo una vez al día, con la oportunidad de mejorar sus modelos en base a los resultados.

3.1.2 Descripción del dataset

El dataset proporcionado para la fase de entrenamiento se encuentra dividido en dos partes: la parte de entrenamiento y la parte de validación. Ambos datasets tienen características similares. Se trata de series temporales del dato del acelerómetro de helicópteros de Airbus medido en testeos en diferentes localizaciones y vuelos. Por otro lado, una misma secuencia se mide desde tres distintos ángulos (X, Y, Z), lo que da lugar a tres secuencias con características similares y diferencias dadas por el método de medición. Puede ser importante mencionar que dichos ángulos pueden no ser los mismos en el dataset de entrenamiento y en el de validación, y lo mismo ocurre con las localizaciones.

La clasificación se realiza sobre secuencias, y cada una debe ser clasificada como normal o anomalía sin necesidad de especificar en qué intervalo de tiempo dentro de la secuencia se produce la anomalía. Cada secuencia se compone de 61.440 columnas que corresponden con las medidas del acelerómetro durante un minuto con una frecuencia de 1024 Hz. Todas las secuencias miden lo mismo.

El dataset de entrenamiento consta de 1677 secuencias. Todas las secuencias son consideradas normales, no anómalas, por expertos analistas de Airbus. El dataset de validación tiene 594 secuencias de las cuales se desconoce la cantidad de normales y anómalas.

3.1.3 Evaluación

Como se ha dicho antes, cada día de la fase de entrenamiento los equipos participantes pueden realizar a lo sumo una entrega de la solución de las clasificaciones de validación y obteniendo los resultados de manera inmediata. Los resultados se componen de tres valores.

El primer valor es el *recall.* Se calcula como el número de secuencias anómalas clasificadas como anómalas dividido entre el número total de secuencias anómalas. En otras palabras, verdaderos positivos dividido entre falsos negativos más verdaderos positivos.

El segundo valor es la precisión, que se define como los verdaderos positivos dividido entre los verdaderos positivos más los falsos positivos. Será mas alto mientras menos secuencias normales se clasifiquen erróneamente como anómalas.

Por último, se da el valor $F_\beta score$ que definido por la siguiente fórmula, da mucho más peso a la precisión que al recall, lo que significa que se busca que los modelos generen la menor cantidad posible de falsos positivos.

$$F_\beta score = \frac{(1 + \beta^2) \times precission \times recall}{\beta^2 precission + recall}, \beta = 0.1$$

3.2 Herramientas utilizadas

A lo largo del proyecto se ha utilizado exclusivamente el lenguaje de programación Python para construir los modelos de clasificación. Esta elección se basa en la utilización de este lenguaje mayoritariamente en esta área de la ingeniería informática y por ello, resulta más fácil tanto su desarrollo como su investigación. El lenguaje python se caracteriza por poner como objetivo la legibilidad del código a la hora de especificar su sintaxis. De esta manera se simplifica bastante, siendo conocido como uno de los lenguajes de programación con una curva de aprendizaje más corta. Por otro lado, es un lenguaje interpretado, lo que quiere decir que no necesita de compilación.

Al ser el lenguaje más usado en la ciencia de datos, se facilita la construcción de modelos con librerías de libre uso y código abierto optimizadas para tener el mejor rendimiento posible. A continuación, en los siguientes puntos se describen las librerías y herramientas utilizadas a lo largo del proyecto.

3.2.1 NumPy

Una de las extensiones de Python más importantes en el estudio de datos es NumPy. Esta librería aporta soporte para todo tipo de operaciones entre vectores y matrices. Permite realizar operaciones de alto nivel simplificando notablemente el código, teniendo implementaciones internas de los algoritmos altamente optimizadas. En general, su funcionalidad se puede comparar con MATLAB ya que ambos son interpretados y permiten operaciones de alto nivel con arrays y vectores. Numpy está complementada con otras librerías, como Scikit-learn y Matplotlib.

3.2.2 Scikit-learn

Scikit-learn es una librería de software libre para machine learning en Python. Funciona en paralelo con la librería de NumPy para realizar el manejo de los datos. Es una de las librerías más usadas en el estado del arte a la hora de utilizar algoritmos de clasificación, regresión y clustering como SVM, random forests, k-nn y DBSCAN. A lo largo del proyecto se utilizará en varias situaciones, principalmente para el algoritmo de k-nn, aunque también se realizarán pruebas con otros algoritmos de clasificación.

3.2.3 Matplotlib

A la hora de generar gráficas de vectores en Python, Matlplotlib es la librería más usada. Además, es complementaria a la librería NumPy. Se usará a lo largo del proyecto para analizar los datos, modelarlos y analizar los resultados de cada modelo fácilmente.

3.2.4 TensorFlow

TensorFlow es una librería de código abierto multiplataforma desarrollada por el equipo *Google Brain* inicialmente considerada solo para el uso ineterno en Google. No se limita a Python y está disponible en otros lenguajes como C++, Java, R y JavaScript entre otros. Está centrada en el desarrollo de machine learning y, principalmente, es el framework más aplicado para el uso de algoritmos deep learning. Los modelos de deep learning son fácilmente implementados en TensorFlow con la ayuda de Keras.

3.2.5 Keras

Keras es una API de alto nivel para la construcción de redes neuronales y, en especial, redes convolucionales y redes recurrentes. Ha sido diseñada poniendo como prioridad principal la experiencia del usuario y la facilidad para utilizarla, minimizando las acciones necesarias para construir modelos y proporcionando información clara sobre los errores de ejecución. Perminte construir redes neuronales de manera muy modularizada, incluyendo la representación de capas de las redes, funciones de activación, optimizadores y funciones de coste como módulos fáciles de incluir y modificar en cualquier modelo. Junto con TensorfFlow, es clave para la construcción de modelos de redes neuronales en este proyecto.

3.2.6 Distribución Anaconda

Anaconda es una distribución de código abierto, tanto para Python como para el lengueje de programación R, para la implementación de modelos de ciencia de datos, machine learning

y procesamiento de datos. Su objetivo es simplificar el manejo de los paquetes y el despliegue de las aplicaciones. En particular, en el proyecto se hace uso de Jupiter Notebook, que consiste en un documento JSON que contiene una lista de celdas las cuales pueden contener código y texto entre otras cosas. Este notebook facilita la ejecución de las implementaciones de modelos, así como su despliegue en diferentes entornos. Además, al dar la posibilidad de escribir texto e incluir gráficas, sirve de plataforma para realizar pruebas fácilmente y así como analizar los resultados rápidamente.

3.2.7 Google Colab

Hoy en día la profesión de ingeniería de informática tiene la facilidad de poder ser practicada sin necesidad de utilizar máquinas que estén fuera del alcance de la mayoría de las personas. Sin embargo, en el area de la ciencia de datos muchas veces se da la excepción, al tener que entrenar modelos con varios GBs de datos y miles o millones de parámetros que ajustar, lo cual suele ser demasiado para un ordenador personal tradicional. *Google Colab* aporta una solución para desplegar modelos de aprendizaje automático de un tamaño considerado mediano. Consiste en un servicio en la nube basado en los Jupiter Notebooks mencionados en el punto anterior y que además permite usar una GPU de manera gratuita y con prestaciones excepcionales. Por otro lado, soporta el desarrollo de modelos de aprendizaje automático usando las librerías más populares, incluyendo todas las que se han mencionado en los anteriores apartados. A lo largo de este proyecto se utiliza Google Colab para entrenar algunos de los modelos que se han desarrollado y tienen un entrenamiento particularmente costoso [11].

3.3 Proceso de trabajo

El desarrollo del proyecto se compone de varios pasos más o menos separables que corresponden con los pasos que generalmente se toman en la mayoría de los proyectos de la misma área. Estos pasos siguen un modelo iterativo, es decir, es una secuencia de fases que se repiten una vez terminadas hasta conseguir el modelo o los modelos que mejor resuelvan el problema, cumpliendo todos los requisitos necesarios. Estas iteraciones comienzan con el análisis de los datos y terminan con la entrega de la solución nueva del modelo y el análisis de los resultados. El proyecto, como se ha mencionado antes, consiste en clasificar un dataset de validación. Y se permite entregar una solución diaria, lo que significa que se tiene los resultados del modelo construido de manera diaria. Estos modelos luego son entrenados con el dataset de entrenamiento, el cual se compone de secuencias clasificadas como normales. Dicho esto, los pasos son los siguientes.

3.3.1 Análisis de los datos

El primer paso de una iteración consiste en el análisis de los datos. Como en cualquier proyecto de ciencia de datos, se necesita saber la forma, distribución y claridad de los datos. Esto conlleva realizar gráficas, en este caso graficas dependientes del tiempo, realizar algunos análisis estadísticos como la media o la desviación típica y modelados de señales del estado del arte. Por otro lado se suelen realizar comparaciones entre cada ejemplo del dataset para saber qué tan diferentes son cada uno de los ejemplos entre ellos.

Si anteriormente ya se ha hecho una entrega de un modelo anterior, este análisis se realiza con un objetivo que viene de la vista de los resultados de la entrega de la solución.

3.3.2 Elección e implementación del modelo

Con una idea de qué es lo que se quiere conseguir, sea un modelo que detecte más anomalías, un modelo que detecte menos para obtener una mejor precisión en los resultados, o un modelo que detecte un tipo en particular de anomalías que se pueda usar en conjunto con otros, se comienza la fase de elección e implementación. Se elige un modelo que sepa diferenciar los detalles de las secuencias que se consideran importantes para dicho objetivo.

Con el modelo elegido, la siguiente tarea consiste en comenzar la implementación a la vez que se van añadiendo los parámetros que necesite este modelo. Estos parámetros pueden ser seleccionados en base a una investigación previa sobre el area en el estado del arte, en base al objetivo que se quiere conseguir con este modelo, o simplemente se elige un set de distintos parámetros con los que se harán distintas pruebas para buscar la solución óptima.

3.3.3 Entrenamiento y validación

Esta fase, como bien dice el título, comienza con el entrenamiendo del modelo. Una vez entrenado la primera vez, se valida para confirmar su buen funcionamiento y se analiza, por ejemplo con una matriz de confusión, para entender sus fortalezas y debilidades. Al solo poder hacer una entrega diaria de la clasificación de validación, esta validación previa del modelo se realiza en base a soluciones previas enviadas. Cada entrega proporciona información sobre los resultados de la clasificación y permite realizar comparaciones de soluciones de otras pruebas para predecir si el modelo no es tan bueno o si vale la pena validar con la entrega diaria.

3.3.4 Obtención y análisis de resultados

Finalmente, tras haber generado una solución con el modelo, se entrega para obtener los resultados que representan la validación final. Con estos resultados, que contienen el recall y la precisión, se sabe cuántos verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos tiene el modelo, y comparando estos resultados con entregas previas, se puede saber si realmente ha habido una mejora notable. En esta fase se analizan los resultados para decidir si seguir utilizando el modelo con algunos cambios o directamente buscar otro modelo que funcione mejor. Estos cambios pueden ser de dos tipos: cambios en los atributos que representan cada secuencia a clasificar o cambios directos en la arquitectura del modelo, como puede ser el número de capas de una red o el umbral a partir del cual se consideran las secuencias como anomalías. Finalmente, se termina esta iteración y se comienza la siguiente, sea con el análisis de los datos en busca de un nuevo modelo, o realizando cambios en el modelo existente.

4 Desarrollo

4.1 Análisis inicial de los datos

El primer paso a la hora de enfrentar un problema de la ciencia de datos es el análisis de estos. Nuestros datos consisten en secuencias temporales del acelerómetro de los helicópteros de Airbus. El sensor mide este dato con una frecuencia de 1024 Hz y cada secuencia del dataset consiste en registros del dato durante un minuto. Es decir, 60 segundos por 1024 Hz, son 61440 mediciones por cada secuencia, o columnas por cada individuo a clasificar del dataset. Al representar estas secuencias dependiendo del tiempo, se puede ver en la figura 4-1 su aspecto más general.

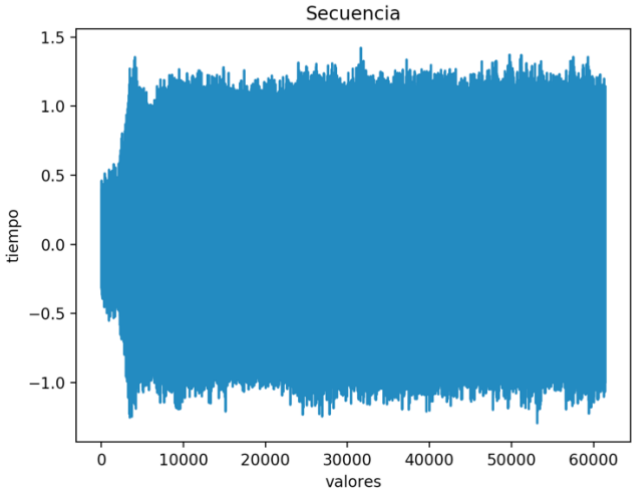

Figura 4-1: Gráfica de secuencia

La gráfica anterior no nos da mucha información al tener tantos valores. Básicamente lo que se puede extraer es que encontramos valores en un intervalo entre [-1, 1] y con una media, seguramente cercana al cero. Al observar una gráfica concentrada en un intervalo más pequeño de solo unas pocas mediciones, se percibe claramente que se trata, básicamente, de una señal oscilatoria. En la figura 4-2 se ve la forma clara de una señal que oscila alrededor del cero y tiene una amplitud de cerca de uno. Este es solo un ejemplo de una secuencia del dataset, pero aunque haya algunas variaciones, todas las secuencias miden el mismo dato y se trata siempre de una señal oscilatoria.

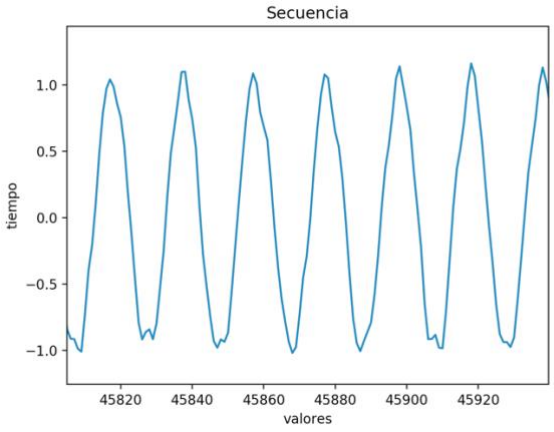

Figura 4-2: Gráfica de intervalo de una secuencia

Se sabe que estamos tratando señales oscilatorias, pero los valores y el intervalo que recorren puede variar de una secuencia a otra. Para tener una idea general de cómo son todas las secuencias y para intentar percibir alguna tendencia o patrón, realizamos una gráfica que las compare. Esta gráfica consiste simplemente en mostrar la media, valor mínimo y valor máximo de cada una de las secuencias del dataset de entrenamiento. Ya que se trata de 1677 secuencias, se construyen 8 gráficas con 200 secuencias cada una, excepto la última. Así se intenta evitar perder los detalles en una gráfica con todas las secuencias juntas. A continuación se presentan dos ejemplos de estas 8 gráficas.

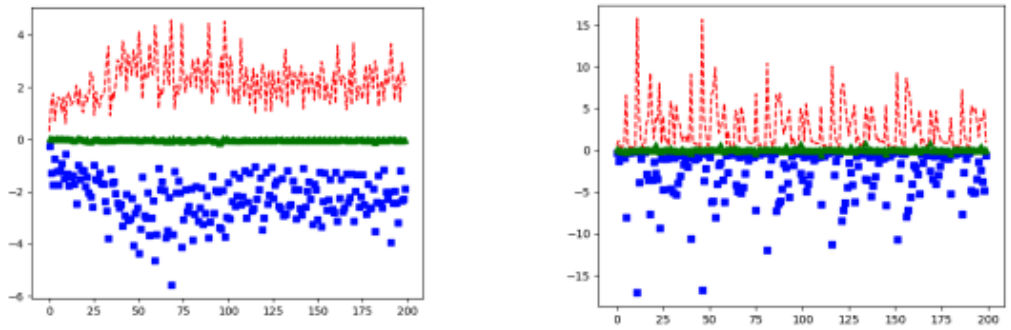

Figura 4-3: Gráficas de media, máximo y mínimo

Se puede apreciar que la media es bastante constante en el valor cero, cosa que se repite en todas las demás gráficas. Acerca de los valores máximos y mínimos de cada secuencia, se han escogido las dos gráficas más dispares. La de la izquierda, correspondiente con las primeras 200 secuencias, tiene como valor máximo y mínimo ± 4 y la de la derecha, con los valores más extremos, tiene ± 15. Es importante aclarar que los datos no explican o representan nada, ya que han sido modificados por Airbus antes de entregarlos a los participantes sin exponer dichas modificaciones. Por lo tanto, no se deben realizar conclusiones precipitadas que no se basen más que en la comparación con las secuencias que nos informan como normales.

4.1.1 Análisis espectral

Una vez identificados los datos como secuencias de señales oscilatorias, se procede a hacer un análisis que parece apropiado para este tipo de señales. En el estado del arte del modelado de señales temporales de este tipo se utilizan los espectrogramas para descomponer la señal en sus partes más simples y así tener no solo más información, sino también información que facilite entender e identificar patrones. Se ha tomado la decisión de hacer un análisis espectral tras analizar los espectros de las señales de entrenamiento y observar una gran regularidad en la representación espectral de estas señales, por lo que esperamos que esta representación sea apropiada para que los subsiguientes modelos logren captar las diferencias entre los individuos de cada clase con facilidad.

El espectrograma termina siendo una imagen, o una matriz de valores, que representan los espectros de la señal en intervalos de tiempo. Esto proporciona una imagen con las amplitudes en función de la frecuencia y el tiempo. Utilizando la librería Matplotlib se

consigue calcular el espectrograma de una secuencia fácilmente, teniendo como resultado la figura 4-4.

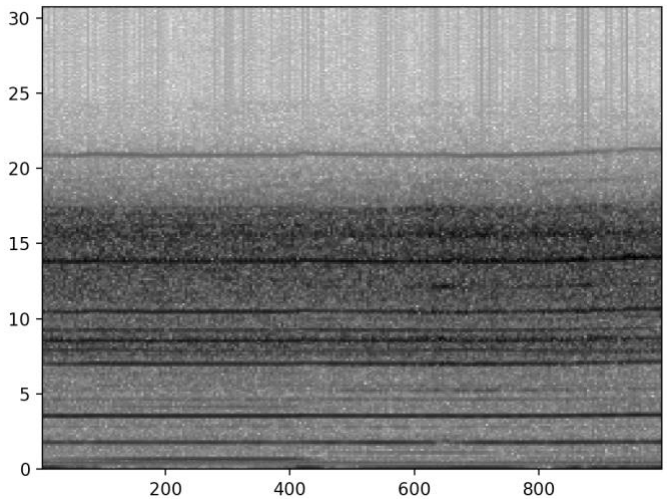

Figura 4-4: Espectrograma

Observando el espectrograma de ejemplo, se perciben las frecuencias más importantes claramente como líneas horizontales. Por ejemplo, la línea oscura de menor frecuencia está en la frecuencia cercana a $y = 2$Hz. El resto de línas horizontales representan los armónicos de la frecuencia principal. La mayoría de las secuencias del dataset de entrenamiento tienen un espectrograma muy similar al del ejemplo. Como se ha mencionado antes, las secuencias están representadas desde tres ángulos distintos, es decir, que hay 1.677 secuencias, pero realmente hay 559 distintas medidas desde los tres ángulos. Un ejemplo de un set de tres secuencias iguales es el siguiente.

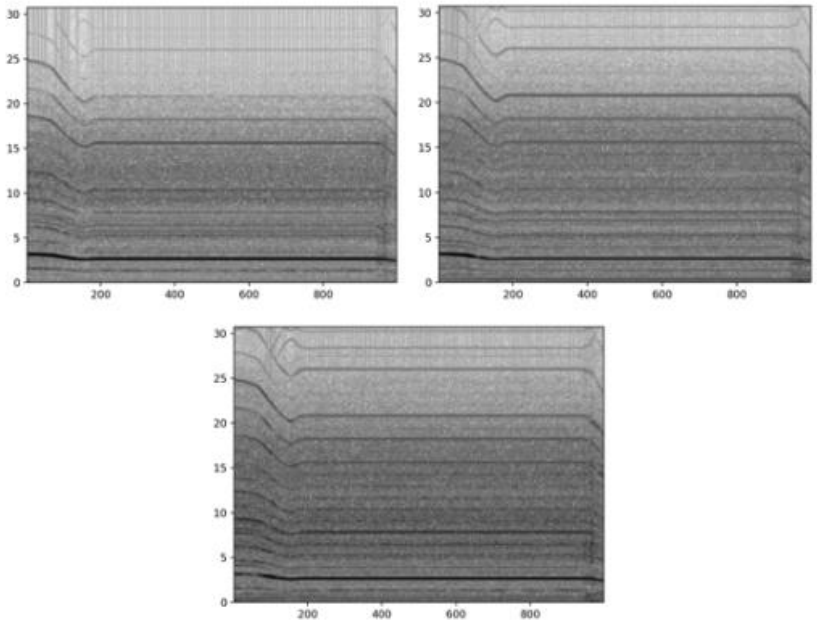

Figura 4-5: Secuencia desde sus tres ángulos

A pesar de que son muy parecidos, estos tres espectrogramas contienen valores con diferencias importantes. Se pueden ver diferencias, por ejemplo, en la parte superior de la primera secuencia que contiene lo que parecen líneas verticales solapadas. O entre las

rectas $y = 4$ e $y = 10$ se oscurece más la secuencia de abajo, lo que indica que tendrá valores más altos en la escala de grises.

Tras pensar en cómo realizar el primer modelo entrenado únicamente con secuencias normales (no anómalas), se decide por empezar una primera aproximación con un k-nn.

4.1.2 Primera aproximación, k-nn

El objetivo de este primer modelo es comparar cada secuencia del conjunto a clasificar con las secuencias que se consideran normales para determinar su semejanza con un grado. De esa manera se podrán clasificar como anómalos aquellos individuos con un grado anormalmente alto.

El k-nn consiste en calcular la distancia al conjunto de secuencias más cercanas según una métrica que mide la semejanza entre secuencias. La suma de estas distancias puede representar el grado de rareza de una secuencia. Comparando cada una de las secuencias de validación con todas las de entrenamiento y realizando la suma de las k distancias más pequeñas de estas secuencias, se obtiene el grado de anomalía de cada una. Esta es la primera aproximación sencilla para identificar las primeras secuencias anómalas con la idea de realizar un primer modelo para observar su funcionamiento y con unos resultados, conocer mejor el problema al que nos estamos enfrentando.

4.1.2.1 Implementación

La implementación es la siguiente. La representación de cada secuencia debe contener información de gran importancia pero intentando evitar cantidades muy grandes de atributos, ya que el k-nn no funciona bien en tales casos. La información se busca en el espectrograma. Cada espectograma es representado con una matriz de tamaño $257x239$ dando en total 61423 atributos. Cada uno de estos atributos es el valor en la escala de grises para determinar el color en ese pixel de la imagen. Ya que estos atributos son demasiados para nuestro modelo, se busca la manera de comprimirlos. Esta cantidad de atributos es demasiado grande para un modelo como el de k-nn, ya que el cálculo de la distancia puede no diferenciar entre ejemplos de manera correcta con tantos valores. Se procede a analizar los espectrogramas para buscar una manera de comprimir la información. Viendo los espectogramas, como los mostrados en las figuras anteriores, la mayoría de la información viene dada por líneas horizontales. Mientras que, por otro lado, en las columnas no se obtiene información muy determinante. Se decide realizar una compresión de los atributos a partir de esta idea, calculando las medias horizontales de la matriz. Al final, haciendo dichas medias, se obtiene un vector de 257 elementos, lo cual es un tamaño mucho más conveniente para nuestro modelo.

Una de las razones por las cuales se utiliza este vector viene por la observación de tendencias en los espectogramas del dataset de entrenamiento. Observando bastantes espectrogramas, estos parecen tener las frecuencias predominantes en un conjunto pequeño de valores. Como se puede ver en la figura 4-4, la frecuencia principal tiene un valor cercano a 2 Hz, mientras que el primer armónico (cercano a 4Hz) suele ser el predominante en amplitud, lo cual se repite para la mayoría de las secuencias del dataset junto con otros valores también habituales. En el datset de validación, en cambio, no se encuentran dichas tendencias de manera tan habitual, obteniendo valores de frecuencias predominantes inusuales respecto a los del entrenamiento. Este patrón da a entender que pueden deberse a las anomalías. Por ello, un vector de medias horizontales representa bastante bien la información que, por ahora, se entiende por importante.

4.1.2.2 Creación del dataset

Una vez definidos los atributos que representan las secuencias, es necesario crear un dataset nuevo con tales atributos. Esta tarea de crear el dataset se repetirá a lo largo del proyecto cada vez que sea necesario realizar modificaciones de la representación de los individuos. En el caso actual, los atributos están formados por el vector de medias horizontales de la matriz del espectrograma. Calcular este vector de medias supone, no solo calcular las 257 medias por cada secuencia, sino calcular toda la matriz del espectrograma con anterioridad. Estos cálculos son relativamente complejos y realizando pruebas, la creación del dataset completo tarda unos 5 minutos. Debido a que no es eficiente esperar estos minutos cada vez que se haga una prueba del modelo, el cual es mucho más rápido, se decide separar completamente la creación del dataset a un módulo aparte para que su ejecución se realice solo una vez. Este módulo, básicamente, lee los ficheros proporcionados por Airbus directamente, realiza los cálculos necesarios para crear el nuevo dataset, y guarda el dataset con el formato CSV necesario para que pueda ser eficientemente cargado con funciones optimizadas de la librería NumPy.

Esta creación de un módulo beneficiará numerosas veces a lo largo del proyecto para facilitar la aplicación de pequeños cambios en los atributos de las secuencias, como añadir una normalización, o grandes cambios que modifique totalmente la forma de los datos.

La normalización, paso importante en todo modelado de datos, no es necesaria en este caso, ya que todos ellos provienen del espectrograma, el cual ya está normalizado de por sí, ya que las señales originales también estaban normalizadas en media y en varianza por Airbus. Es decir, todos los elementos del vector están calculados y medidos de la misma manera, que es todo lo que necesita un k-nn para funcionar correctamente.

4.1.2.3 Parámetros del modelo

El número de vecinos elegido es 5. Sabiendo que cada secuencia está repetida 3 veces (con pequeñas diferencias) se desea evitar que las tres primeras distancias a un individuo de validación coincida con los tres ángulos de una misma secuencia. Para ello, se usa un k mayor que 3. También conviene que el número de vecinos no sea demasiado alto. Este número de vecinos, una vez entregada la primera solución, se optimizará probando con otros k en busca de la mejor solución. Tras calcular las distancias y sus sumas para cada secuencia, se ordenan y se representan en una gráfica, obteniendo las gráficas mostradas en las figuras 4-6 y 4-7. La primera gráfica corresponde con el resultado de aplicar el algoritmo al conjunto de validación y la segunda al conjunto de entrenamiento, el cual se supone que solo contiene ejemplos de la clase normal.

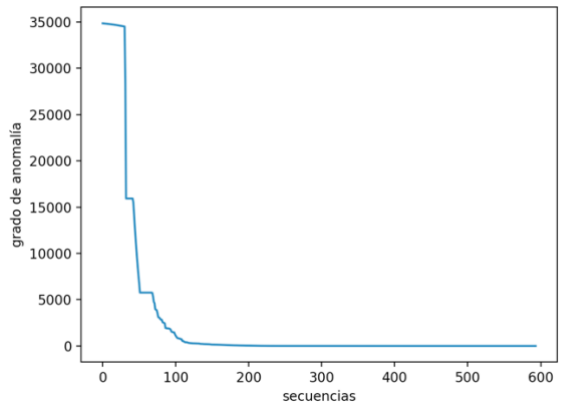

Figura 4-6: Resultado k-nn, k=5 en el conjunto de datos de entrenamiento

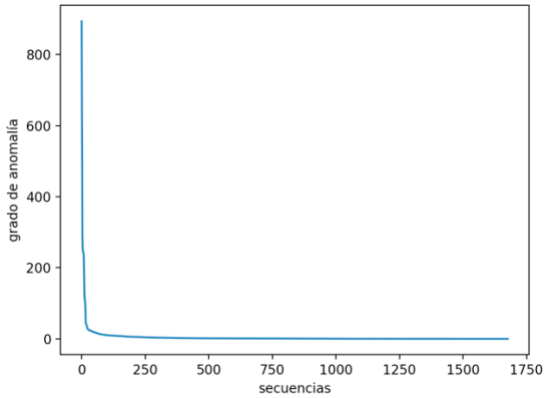

Figura 4-7: Resultado k-nn, k=5 en el conjunto de datos de validación

Las secuencias de validación, a partir de la 100, bajan su grado de anomalía notablemente. Ahora llega el momento de decidir qué umbral seleccionar. Para ello, se hace uso de ambas gráficas. Este umbral, finalmente, es 1128.962334, número que corresponde a la secuencia más anómala en la gráfica del dataset de entrenamiento, ya que esta secuencia, según el enunciado del concurso, es considerada de clase normal. Se clasifican, casualmente, 100 secuencias como anómalas con este umbral. El resultado al entregar esta solución se muestra en la siguiente tabla (4-1).

Primer resultado k-nn	
Recall	0.3367003367003367
$F_\beta score$	0.860096267655646
Precisión	1

Tabla 4-1: Primer resultado k-nn

4.1.2.4 Conclusiones sobre el primer modelo

Los resultados son notablemente buenos, ya que se obtiene precisión perfecta, lo que significa que todas las secuencias clasificadas como anómalas son verdaderos positivos. Un análisis más detallado de los resultados permite conseguir el número de secuencias anómalas y secuencias normales que hay en el conjunto de datos de validación. Con el valor dado del recall y sabiendo que tenemos 100 secuencias clasificadas como anómalas, podemos obtener el número de secuencias anómalas de la siguiente manera.

$$recall = \frac{100}{100 + FN} = 0.3367; \quad 100 + FN = \frac{100}{0.3367} \approx 297$$

De esta manera sabemos que el número secuencias anómalas es de 297 y lo mismo para las normales. Se puede ver que aún quedan muchas secuencias por clasificar como anomalías pero, teniendo precisión perfecta, seguramente se pueda mejorar estos resultados fácilmente. Por último, se puede apreciar un favoritismo por buena precisión en el valor calculado para el $F_\beta score$, ya que a pesar de que el recall es malo, el valor es bastante alto. Se entiende que, para los intereses de Airbus, tener falsos positivos es altamente perjudicial en la monitorización de fallos en los sistemas de los helicópteros. Esto nos puede ser de beneficio, para así obtener altos valores de $F_\beta score$ sin necesidad de tener un acierto general tan alto.

4.1.3 Ajustes y optimización del modelo con k-nn

Los primeros resultados obtenidos indican que el modelo tiene bastante calidad. Estos resultados se deben principalmente a la elección de unos buenos atributos para cada secuencia que permitan entender la forma de las secuencias normales para poder diferenciarlas de las anómalas. Además, el modelo no ha tenido problemas en que cada secuencia albergue más de 200 atributos, lo cual en algunos casos puede ser un problema de alta dimensionalidad. Al ver que este primer intento ha funcionado bien, a continuación se desarrollan ajustes en el modelo junto con optimizaciones para buscar el k-nn más óptimo posible realizando arreglos en los parámetros del modelo así como en los atributos que forman las secuencias.

4.1.3.1 Búsqueda del umbral óptimo

En los resultados anteriores se ha obtenido una precisión 1 con 100 secuencias clasificadas como anomalías y un umbral de 1128.962334 seleccionado por ser el valor de la secuencia normal con mayor grado de anomalía. Basándonos en la asunción de que puedan existir secuencias anómalas en el dataset de entrenamiento, a pesar de que en el enunciado del concurso se anuncie que en principio, todas son normales, se intenta buscar un umbral menor al anterior. Esta asunción se basa también en que se desee un modelo que pueda entrenar con pequeños fallos en el datset, es decir, unos pocos ejemplos clasificados erróneamente. La tolerancia a fallos suele tener cierta importancia en este tipo de problemas.

Este umbral óptimo debe tener las siguientes características:

- Tener precisión 1. Lo que se busca es el valor óptimo de la métrica $\boldsymbol{F_\beta score}$ la cual perjudica considerablemente el modelo si su solución contiene falsos positivos.
- Tener el mayor recall posible. Es decir, se busca que al bajar el umbral en lo más mínimo, la precisión deje de ser 1.

El primer paso es el análisis de la gráfica mostrada en la figura 4-7 sobre el dataset de validación. Se realiza una búsqueda de un posible escalón que indique que las secuencias mayores sean potencialmente anómalas y las menores no. Aparte de analizar esta gráfica, se analizan visualmente los espectrogramas en busca de indicios de normalidad o rareza. Después de haber visualizado cuantiosas secuencias del dataset de entrenamiento, es fácil identificar secuencias con altas probabilidades de pertenecer a la clase positiva.

Se analizan las secuencias con grados de anomalía menores de 100 hasta los valores alrededor de 100. Se observan los espectogramas y se confirma que muchas de estas secuencias tienen cualidades excepcionales que, a juicio del ojo humano, se considerarían anómalas. Un ejemplo de uno de estos espectrogramas es el siguiente.

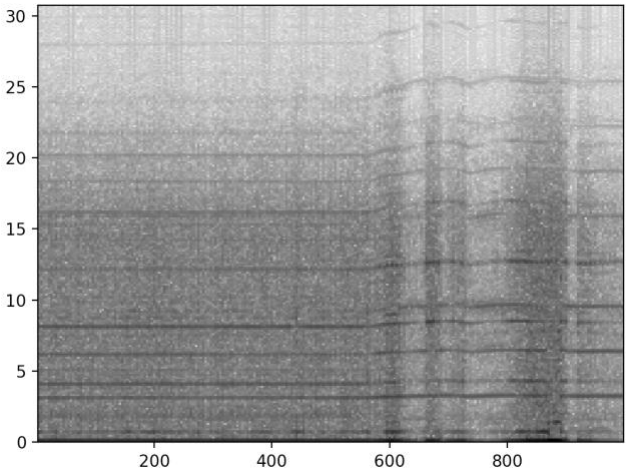

Figura 4-8: Espectrograma de secuencia potencialmente anómala

Este espectrograma corresponde con la secuencia de id 458 y tiene un grado de anomalía de alrededor de 150. La razón de ser potencialmente anómala es que tiene la línea más oscura cercana al 0. Esta característica es inusual en el dataset de entrenamiento y se da con abundancia entre las secuencias que ya se conocen como anómalas. Esta cualidad se da no solo en esta secuencia, sino en muchas con valores parecidos en su grado de anomalía. Debido a ello, se intenta buscar un umbral que las clasifique todas como anómalas, y de esta manera se conseguiría aprobar esta teoría.

Buscando el umbral en la gráfica de secuencias y sus grados de anomalía, se decide usar el umbral de 58.834819. Este valor divide la gráfica en dos, precisamente alrededor de un escalón en la gráfica que puede representar el comienzo de la clasificación de falsos positivos por no poder diferenciar bien sus cualidades. Se intenta evitar esta clasificación colocando el umbral anteriormente a la zona de la gráfica donde la curva pierde más pendiente y comienza a estabilizarse.

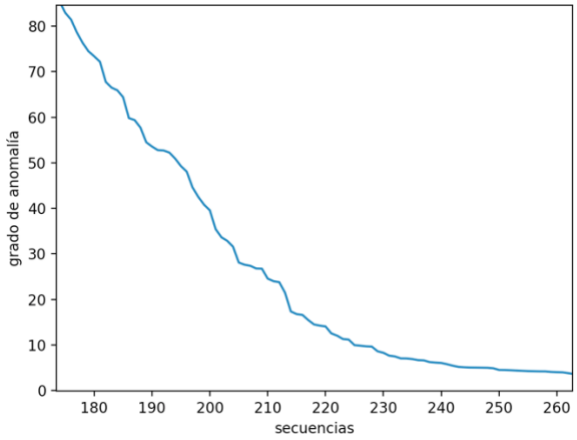

Figura 4-9: Detalle de la gráfica en la zona posterior al umbral

Colocando el umbral en dicho valor y entregando la segunda solución al reto se obtienen los siguientes resultados.

Segundo resultado k-nn	
Recall	0.632996632996633
$F_\beta score$	0.9543147208121828
Precisión	1

Tabla 4-2: Segundo resultado k-nn

Como se puede observar, se sigue viendo un perfecto funcionamiento del modelo. Esta vez superando incluso el 0.90 en $F_\beta score$ manteniendo la precisión en 1. Para decidir el siguiente umbral a evaluar en validación es necesario analizar bien los datos que se conocen para reducir el número de pruebas hasta conseguir el umbral óptimo, ya que solo se puede entregar una solución diaria según las reglas del reto.

En el umbral anterior, se clasifican 188 secuencias como anómalas. Esta vez se tiene una cantidad importante de secuencias anómalas, aunque aún lejos de la cantidad total de 297. Para buscar el umbral óptimo se necesita saber realmente qué tan bueno es el modelo. Mientras mejor sea el modelo, más bajo será el umbral. Debido a la incertidumbre acerca de la cantidad de secuencias que el modelo es capáz de clasificar como anómalas sin perjudicar la precisión, se decide seleccionar el umbral que clasifique todas las secuencias, las 297, como anomalías. De esta manera se sabrá si aún hay muchas secuencias que se pueden clasificar como anomalías además de las 188 de la prueba previa. Este umbral, finalmente, es de 2.07. Este valor es bastante bajo y se sabe que es altamente probable la aparición de falsos positivos. Pero el objetivo de esta prueba es la de conocer la cantidad de falsos positivos. Los resultados son los siguientes.

Tercer resultado k-nn	
Recall	0.8417508417508418
$F_\beta score$	0.8417508417508417
Precisión	0.8417508417508418

Tabla 4-3: Tercer resultado k-nn

Como se puede ver, el resultado no es muy bueno. De las 297 secuencias clasificadas como anómalas, realizando los cálculos ya mencionados vemos que solo hay 250 verdaderos positivos. El siguiente paso es el de intentar eliminar tantos falsos positivos y se enfoca aumentando el umbral a 4.9 para obtener 250 anomalías. Los resultados de esta última prueba son los siguientes.

Cuarto resultado k-nn	
Recall	0.797979797979798
$F_\beta score$	0.9335091966899144
Precisión	0.948

Tabla 4-4: Cuarto resultado k-nn

Los valores mejoran y se obtienen tan solo 13 falsos positivos. Todas estas pruebas sirven de métricas para evaluar los ajustes en el modelo k-nn que se describen en el siguiente apartado.

4.1.3.2 Ajustes de los parámetros

El primer ajuste consiste en limpiar el dataset de entrenamiento. Al haber obtenido precisión 1 con un umbral mucho menor al grado de anomalía de varias secuencias de entrenamiento, se puede considerar la eliminación de estas secuencias en el entrenamiento del modelo por contener errores. Para ello se usa un umbral que clasifica 50 secuencias como anómalas. Tras identificar estas secuencias se eliminan del dataset de entrenamiento.

Sin embargo, al volver a entrenar el modelo con el dataset limpio, el resultado es prácticamente el mismo. No solo clasifica las mismas 188 secuencias con el umbral 58.834819 sino que los grados de las secuencias son los mismos. Por otro lado, solo hay 2 clasificaciones distintas al colocar el umbral en 297 secuencias, lo que significa que al modelo no le beneficia la limpieza de los datos.

4.1.3.2.1 Ajuste del número de vecinos

Anteriormente se explica la razón de elegir un k=5. En este apartado se realizan pruebas para buscar el k óptimo. La manera de validar si una k obtiene mejores resultados, consiste en compararlo con el modelo con k=5. Se sabe que que con k=5 y clasificando 297 anomalías se obtienen 47 falsos positivos y que 13 de ellos están en secuencias con grado menor a 4.9. De tal manera que para que un modelo sea significativamente mejor, debe tener secuencias clasificadas diferentemente y deben estar en posiciones menores a 250. Si al colocar el umbral en 4.9, el modelo con k distinto de 5 no tiene muchas diferencias con k=5, entonces el modelo no se va a considerar mejor.

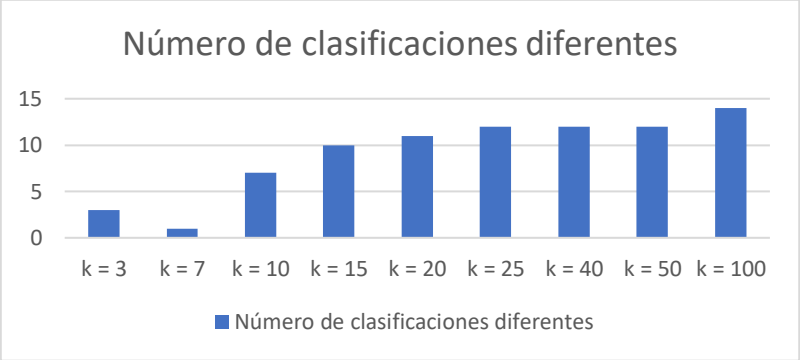

Figura 4-20: Gráfica de resultados de variar k

En la figura 4-10 se miden el número de clasicaciones diferences entre un k determinado y k=5. Las diferencias se miden entre las 297 secuencias más anómalas. Como se puede ver, las diferencias no se acercan a los 47 falsos positivos que obtiene nuestro modelo original con cinco vecinos próximos. Finalmente, no se considera cambiar el valor del número de vecinos, no solo por la poca variación del modelo, sino que las diferencias ocurren en secuencias con grados muy altos. Esto significa que, entre las 250 secuencias más anómalas no ocurre diferencia alguna y que el modelo seguirá teniendo el mismo $F_\beta score$ independientemente del número de vecinos.

4.1.3.2.2 Diferentes métricas de distancias

A lo largo de la implementación del modelo se ha hecho un estudio de cuál es la métrica deseada en un problema como este. La distancia euclídea es la elegida desde un principio por ser la más usada con problemas tanto categóricos como numéricos [12]. Sin embargo, se han realizado pruebas para comparar los resultados de otras métricas como distancia Minkowski, Manhattan (caso especial de la distancia Minkowski) y Chebyshev pero ninguna ofrece la misma precisión que la euclídea ya que estas no obtienen precisión 1 con 188 secuencias o más.

4.1.3.3 Cambio de atributos por cada secuencia

En este apartado se le proporciona al modelo diferentes atributos por secuencia para intentar así dar más información o información más relevante. El objetivo de estos experimentos es el de conseguir, por un lado, facilitar la clasificación de más secuencias como anómalas sin perder precisión, y por otro, detectar nuevas anomalías en distintos aspectos de los datos con el fin de obtener un modelo aparte que se use conjuntamente al anterior.

4.1.3.3.1 Tres vectores

Representar una secuencia por las medias verticales del espectrograma tiene una importante ventaja en la compresión de la información. Sin embargo, hay que tener cuidado de no perder datos que puedan proporcionar un patrón por el cual clasificar una secuencia como anómala. Observando los espectrogramas se pueden encontrar muchos cuyas líneas horizontales no sean completamente rectas o constantes y contengan variaciones que perjudiquen la compresión de la información en un solo vector de medias. Un ejemplo de ello se muestra en la siguiente figura.

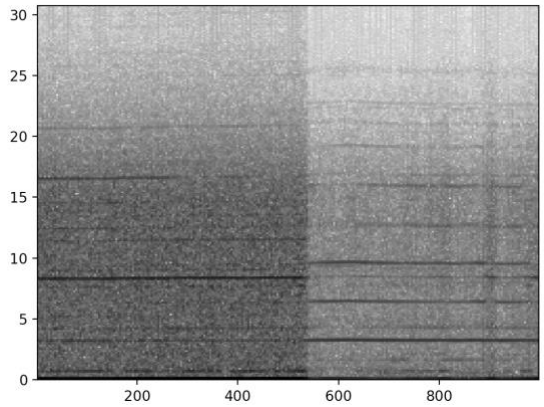

Figura 4-13: Detalle de la gráfica en la zona posterior al umbral

Para perder algo menos de información sobre la variación temporal del espectro, y así proporcionar al modelo datos que puedan mejorar los resultados, se realiza el experimento de generar tres vectores de medias en lugar de solo uno. Cada vector de medias va a representar un tercio de la imagen del espectrograma intentando reproducir la información de su sector. De esta manera se podrán comparar las secuencias por secciones.

Se crea el nuevo dataset y se calcula el grado de anomalía de cada una de las secuencias para comparar los resultados con nuestro modelo anterior.

Los resultados muestran que la precisión baja notablemente, incluso entre las 188 secuencias con más anomalía, las cuales obtienen precisión 1 en el anterior modelo. Por otro lado, tampoco se clasifican diferentes secuencias como anómalas. Si se clasificasen muchas

secuencias distintas al modelo anterior y fuesen verdaderos positivos, se podrían combinar ambos modelos, pero los verdaderos positivos son los mismos. Por estas razones, no se considera el uso de estos atributos como una ventaja.

4.1.3.3.2 Comparación con el vector medio de entrenamiento

En los análisis de los espectrogramas realizados anteriormente se ha mencionado la tendencia de las secuencias normales del dataset de entrenamiento a tener una frecuencia fundamental en en un valor cercano al 2 Hz y una frecuencia con mayor amplitud en un valor cercano a 4 Hz. Por otro lado, también se ha percibido una tendencia de las secuencias ya clasificadas como anómalas correctamente, a tener frecuencias principales en valores distintos, principalmente en 0 y en otros tres valores. Al ser características predominantes en grandes cantidades de secuencias, se podría generar un modelo especializado en detectarlas. Este objetivo se persigue debido a la existencia de secuencias aún no clasificadas como anómalas, pero que mantienen la misma tendencia que sus compañeras anómalas.

Para percibir estas tendencias se realiza la media de todos los vectores de medias del dataset de entrenamiento. Así se consigue un vector medio de medias de los espectrogramas. Este vector se compara con cada una de las secuencias de validación para obtener, al igual que antes, un grado de anomalía. En este caso, el objetivo es obtener la comparación entre dos vectores. Esta comparación se ha realizado mediante el coeficiente de correlación.

Midiendo la similitud entre las secuencias de validación y el vector, se obtiene un valor por cada secuencia en el intervalo $[-1, 1]$, siendo los cercanos a 1 de secuencias muy parecidas al vector y que, por lo tanto, serán clasificadas como normales. Se ordenan las secuencias de menor a mayor por su grado de similitud y se elige un umbral que clasifica 50 como anomalías con el objetivo de evaluar el modelo.

Resultado con vector medio	
Recall	0.010101010101
$F_\beta score$	0.04261696859
Precisión	0.06

Tabla 4-5: Resultado con vector medio

El resultado termina siendo muy deficiente, teniendo solo tres verdaderos positivos de las cincuenta anomalías clasificadas. Este resultado se debe a la pérdida de información al calcular el vector medio. Mientras se intentaba buscar la moda entre las secuencias, en su lugar se ha conseguido una representación muy vaga de la normalidad en una secuencia.

Estas pruebas concluyen los experimentos basados sobre el modelo k-nn. Al final, este modelo simplemente calcula las diferencias entre individuos con una distancia que no da preferencia a ningún conjunto de atributos, sino que los trata a todos de la misma manera. No hay que olvidar que este modelo ha tenido notables resultados, pero es evidente que no se puede optimizar para conseguir evaluaciones mucho mejores a las ya obtenidas.

4.1.4 Modelo de redes convolucionales

Hasta ahora se ha estado utilizando un modelo de aprendizaje no supervisado. El dataset de entrenamiento proporcionado para el reto solo contiene individuos de una clase, lo que no permite el uso de algoritmos de aprendizaje supervisado convencionales. La posibilidad de crear individuos con la clase anómala no es viable debido a la complejidad de los ejemplos

al tratarse de señales muy variadas y con su anomalía difícil de detectar o de entender a simple vista. En base a estos argumentos se decidió en el anterior punto desarrollar un modelo de aprendizaje no supervisado.

Ahora, sin embargo, ya se han realizado predicciones del dataset de validación y algunas de ellas con perfecta precisión. Esto nos proporciona un dataset con elementos de clases anómalas y permite así desarrollar el primer modelo de aprendizaje supervisado.

4.1.4.1 Elección de atributos

Anteriormente se han estado utilizando vectores de medias horizontales de los espectrogramas. Estos atributos han demostrado buenos resultados con un modelo que no aprende dando pesos a los atributos importantes, lo que da indicios de mejoras importantes si el modelo realmente da prioridad a la información importante. Está claro que habrían mejoras utilizando solo los vectores anteriores, pero también se ha demostrado que esta información está comprimida y no contiene toda la información importante de una secuencia, como se puede ver en la figura 4-11. Una manera de solucionar esto podría basarse en seguir la aproximación anterior sobre calcular tres vectores de medias dividiendo el espectrograma en tres partes iguales. Pero aún así, esto supondría comprimir información que en algunos casos suponga la mala interpretación de los aspectos de un individuo. Si se busca la captura completa de la información del espectrograma para que el modelo aprenda por sí solo a encontrar los patrones más valiosos, se necesita todo el espectrograma completo como atributo de cada secuencia. Esto no supondría ninguna pérdida de información. Como ya se ha dicho antes, un espectrograma tiene la forma 257×239, lo que es igual a 61423 atributos o píxeles. Este número es muy cercano a la cantidad de datos medidos en un minuto en cada secuencia, con lo cual se puede considerar que no se pierde nada de información.

4.1.4.2 Creación del dataset

Como se ha explicado con anterioridad, existe un módulo aparte dedicado únicamente a la creación del dataset para permitir un mejor rendimiento en el entrenamiendo del modelo. Esta vez el programa debe calcular el espectrograma de cada secuencia y guardarlo sin modificarlo. Finalmente, con todos los espectrogramas calculados, se guarda en un fichero con un formato legible para NumPy.

Sin embargo, en el entorno usado surgen problemas a la hora de tratar estas cantidades de datos. Para hacerlo más ligero se ha tenido que reducir el tamaño de la matriz del espectrograma con la técnica pooling. Es pooling, como se comenta en la sección del estado del arte, consiste en una reducción o compresión de la información que contiene una matriz con la intención de mantener los datos importantes en las mismas localizaciones. Es decir, si la imagen contiene líneas horizontales en una sección determinada, estas líneas permanecerán representadas en la salida del algoritmo en sus respectivas localizaciones.

En este caso se realiza un max pooling, es decir, por cada ventana permanece el mayor valor. Se utiliza una ventana de tamaño 2×2 y un *stride* también de 2×2. El *stride* determina cuántas casillas avanza cada vez que se cambia de ventana. Finalmente, como de cada ventana que contiene cuatro valores se mantiene uno, la reducción de la matriz es de un 75%. La salida de la compresión consiste en una matriz de tamaño 129×120 que es igual a 15480 parámetros, que es una cantidad aceptable.

El dataset de entrenamiento que contiene las secuencias normales consta de 1677 secuencias, entre las cuales conocemos varias con altas probabilidades de pertenecer a la clase anómala. Por otro lado, las secuencias anómalas que conocemos son en total 205 y provienen del dataset de validación. Una gran diferencia en la cantidad de individuos de cada clase en el

entrenamiento causa que las redes neuronales tengan una probabilidad a priori de que un elemento sea de una clase u otra con favoritismo por la clase con más individuos. Sabemos que la cantidad de elementos de cada clase en validación es de 297 para normales y lo mismo para anómalas. Un priori con preferencia por clases normales perjudicaría el modelo. Para eliminar este problema se duplican las secuencias anómalas varias veces hasta alcanzar la misma cantidad de elementos que tiene la clase normal. El múltiplo que acerca más el número de anómalas al de normales es el 8. Cada secuencia anómala se encontrará 8 veces en el dataset para tener, finalmente, 1677 secuencias normales y 1640 secuencias anómalas. De esta manera se consigue adaptar el problema a un modelo de aprendizaje supervisado. Hay que destacar que entre las 1677 normales habrá un pequeño número de secuencias anómalas que todavía no hemos logrado localizar con el k-nn.

4.1.4.3 Selección del modelo

Los atributos escogidos para cada secuencia hacen de la elección del modelo una tarea sencilla. Cada secuencia es representada por una imagen de su espectrograma que al final consiste básicamente en una matriz cuyos valores indican el tono en la escala de grises en cada pixel de la imagen. El tratamiento de imágenes ha sido perfeccionado a lo largo del tiempo siempre basándose en un modelo, las redes neuronales convolucionales. Debido al constante uso de redes convolucionales o algoritmos basados en ellas para resolver problemas de aprendizaje automático, se decide utilizarlas para continuar resolviendo nuestro problema.

4.1.4.4 Arquitectura de la red

Las redes convolucionales no constan de reglas bien definidas a partir de las cuales construir una arquitectura para una entrada en concreto. Sin embargo, sí hay tendencias que muestran buen comportamiento. Una de ellas es el uso de ventanas pequeñas y de tamaño impar en las capas convolucionales. Es decir, los tamaños más usados son 3×3, 5×5 y en algunos casos 1×1, que es igual a tratar cada pixel como un parámetro. Tamaños pequeños recogen bien los pequeños detalles de la imagen que ventanas grandes no pueden detectar. Por otro lado, las capas convolucionales se juntan en bloques. La arquitectura, al final, es un conjunto de bloques de capas convolucionales y cada uno contiene una capa final max pool que comprime la salida. Estas tendencias básicas se pueden encontrar en las arquitecturas más usadas en el estado del arte como en AlexNet [13].

Para este proyecto se usa una arquitectura popular que ha demostrado tener buenos resultados en la clasificación del dataset MNIST [14]. Comienza con un bloque que contiene dos capas convolucionales de 32 filtros y un tamaño de ventana de 5×5. Tras estas dos capas se comprime la imagen con una capa max pool de ventana 2×2 y finalmente una capa *dropout.* El *dropout* elimina conexiones entre neuronas de forma aleatoria para mejorar la generalización de la red. De esta manera se minimiza el sobreentrenamiento.

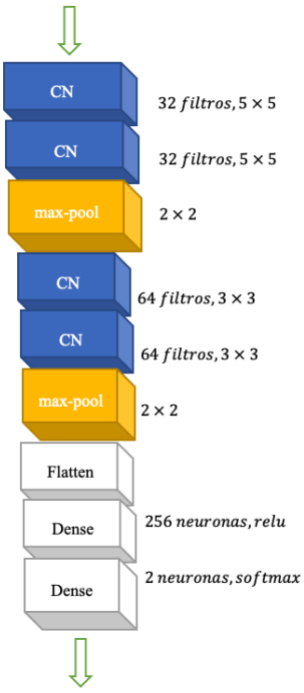

Figura 4-12: Arquitectura de la red

El segundo bloque es parecido, comenzando también con dos capas convolucionales, aunque esta vez contienen 64 filtros y ventanas de tamaño 3×3. A continuación, el bloque consta de una capa max pool con ventanas de 2×2 y *stride* del mismo tamaño, es decir, que reduce en un 75% el tamaño. Al igual que el bloque anterior, este termina con un *dropout*. Tras los dos bloques, la red contiene una capa *flatten* que convierte la matriz de entrada en un vector de una sola dimensión para permitir el uso de capas como las *fully connected*. Las siguientes son una capa *dense* de 256 neuronas y de activación ReLU, una capa de dropout y, por último, otra capa *dense* que contiene dos neuronas, tantas como clases tiene el modelo, y de activación softmax. La activación de softmax final permite obtener como ouput la probabilidad de la secuencia de ser de clase anómala.

4.1.4.5 Parámetros y ajustes

En este apartado se mencionan los detalles adicionales del modelo así como los ajustes de parámetros del modelo. Primero, el dataset se entrena generando de manera aleatoria un conjunto de entrenamiento del 90% y 10% de test. No es conveniente darle demasiado tamaño al test ya que el conjunto de datos no es grande y las características de las secuencias pueden variar notablemente. Por otro lado se utiliza una herramienta de Keras para realizar pequeñas modificaciones en las secuencias como preprocesamiento de imágenes. Se aplican dichas modificaciones seleccionando individuos de manera aleatoria. Estas modificaciones incluyen aplicación de zoom, rotaciones de 10 grados, desplazamiento horizontal y vertical de la imagen.

Se utiliza el algoritmo de optimización RMSprop. Este algoritmo monitoriza las actualizaciones de los pesos que se ajustan en el entrenamiento de la red neuronal con el fin de acelerar y mejorar el aprendizaje de la red. Funciona de manera que si la actualización de un peso es positiva en dos iteraciones seguidas, se entiende que el gradiente tiene una buena dirección y se puede aumentar el ajuste en la siguiente iteración. Si el gradiente cambia de signo, entonces se reducirá el ajuste para conseguir llegar al mínimo de la función de error. Por otro lado, Keras proporciona herramientas para la reducción de la constante de

aprendizaje en caso de que el entrenamiento deje de converger. Esta reducción divide por dos la constante de aprendizaje si no se reduce el error del modelo en tres iteraciones.

Como últimos parámetros del modelo están el número de épocas a 30, y por otro lado, la función de error usada, que es la entropía cruzada categórica. Esta función de error es la óptima para la clasificación categórica. Finalmente se entrena el modelo con estos parámetros y se obtienen los siguientes resultados al clasificar todo el dataset de entrenamiento.

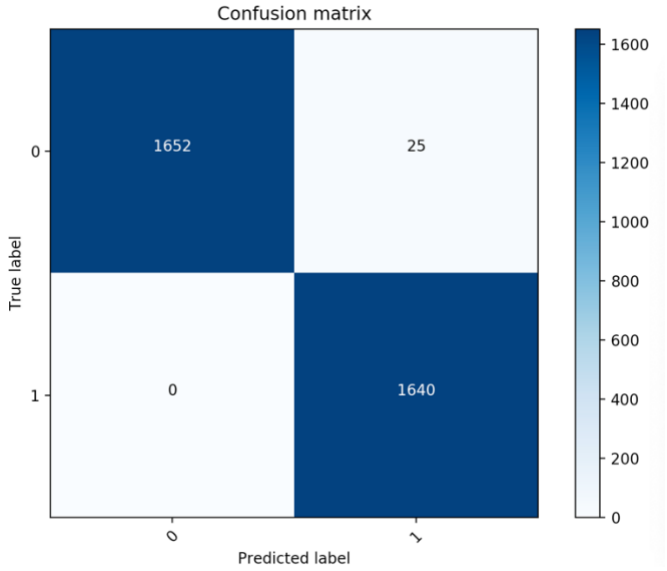

Figura 4-43: Matriz de confusión del dataset de entrenamiento

En la figura 4-13 se representa la clase anómala con 1 y la clase normal con 0. Se puede ver que clasifica 1640 verdaderos positivos, que concuerda con las 1640 secuencias de clase anómala y, por otro lado, detecta 25 secuencias anómalas entre las secuencias supuestamente normales. Esto concuerda con las anteriores conclusiones acerca de la inclusión de errores en el dataset de entrenamiento proporcionado para el reto. El modelo aprende correctamente todas las secuencias anómalas y aparentemente también las normales. Se clasifican las secuencias del dataset de validación, obteniendo 241 anomalías.

Primer resultado cnn	
Recall	0.8114478114478114
$F_\beta score$	0.98117506443
Precisión	1

Tabla 4-6: Primer resultado cnn

El modelo obtiene perfecta precisión, lo cual es ideal para la evaluación del problema. Ahora clasifica 241 secuencias anómalas entrenando con un dataset que contiene 205 anomalías. Es decir, el modelo consigue detectar 36 secuencias más que el anterior. Al conseguir más secuencias anómalas se decide repetir la creación del dataset para entrenar de nuevo el modelo con más ejemplos anómalos que antes, con el objetivo de que aprenda mejor los patrones que distinguen las dos clases.

Esta vez, al tener una cantidad mayor de secuencias anómalas, se multiplican por un factor 7 en lugar de 8 obteniendo así 1687 individuos anómalos. Entrenando el modelo con este nuevo dataset y los parámetros anteriores se obtienen los siguientes resultados.

Segundo resultado cnn	
Recall	0.8552188552188552
$F_\beta score$	0.98621451216
Precisión	1

Tabla 4-7: Segundo resultado cnn

Se detectan 252 anomalías y, una vez más, la precisión es perfecta. El modelo ha mejorado notablemente el recall aunque solo se trate de un aumento de 11 secuencias sobre las 297 anomalías del dataset de validación. La reducción de la mejora en comparación con la anterior se puede atribuir a una mayor dificultad de clasificación de anomalías. Las diferencias entre las secuencias de las dos clases pueden empezar a atribuirse a patrones más difíciles de detectar para el modelo y puede que no sean suficientemente frecuentes en los datos como para que el modelo entienda el patrón. Por ello, se puede considerar esta la inclusión de estas 11 secuencias a verdaderos positivos como una mejora importante.

5 Integración, pruebas y resultados

5.1 Resultados de pruebas con k-nn

El primer modelo desarrollado está basado en el algoritmo de k-nn. Como se ha explicado anteriormente, el algoritmo se entrena con un conjunto de datos que solo contienen individuos de clase normal. Para entrenar dicho modelo se miden distancias entre cada secuencia y se suman las 5 distancias más cercanas para definir el grado de anómalía de cada una de las secuencia del conjunto de datos a clasificar. Este modelo contiene además algunos parámentros y durante el desarrollo se han realizado distintas pruebas realizando pequeños cambios en ellos para obtener así la versión óptima del modelo. A continuación se muestran en una tabla las pruebas cronológicamente realizadas sobre el problema final a solucionar, y que han sido validadas en la plataforma *AirbusAI Gym* de la empresa Airbus.

	$F_\beta score$	P	Recall	acierto	k	umbral	VP	FP	VN	FN
1	0.86	1	0.3367	66.8%	5	1128.962	100	0	297	197
2	0.9543	1	0.63299	81.2%	5	58.834	188	0	297	109
3	0.84175	0.8417	0.84175	84.1%	5	2.07	250	47	250	47
4	0.9335	0.948	0.7979	87.7%	5	4.9	237	13	284	60
5	0.9637	0.9869	0.7643	87.7%	5	8.5	227	3	294	70
6	0.9689	0.9955	0.7474	87.2%	5	12	222	1	296	75
7	0.9699	0.9955	0.7542	87.5%	5	12	224	1	296	73
8	0.962	1	0.6767	84.3%	5	37	201	0	297	93
9	0.042	0.06	0.01	42.3%	-	-	3	47	250	294

Tabla 5-1:Tabla de resultados con k-nn

Las diferencias entre cada una de las pruebas y el análisis de cada resultado con el objetivo de conocer las ventajas y desventajas del modelo se describen en los siguientes puntos que coinciden con los identificadores de las filas de la tabla superior.

1. Esta primera prueba se realiza seteando el umbral al valor de la secuencia más anómala del conjunto de datos de entrenamiento. Para calcular dicho valor se ejecuta el modelo sobre este conjunto y entrenando con el mismo, cambiando la k a 6 vecinos, ya que el más cercano será siempre la propia secuencia y da un valor nulo. La precisión es buena e indica la posible existencia de más verdaderos positivos y el umbral de la función de activación final.
2. Se disminuye el valor del umbral consiguiendo 188 verdaderos positivos y manteniendo la precisión perfecta. Se consigue un acierto aceptable sobre el 80% y el recall mejora notablemente. El modelo demuestra un buen aprendizaje de los patrones que diferencian las clases y ya consigue dar resultados aceptables.
3. Se disminuye aún más el umbral por haber obtenido precisión perfecta en la prueba anterior. Se baja el umbral hasta un valor demasiado bajo lo cual perjudica bastante la precisión y el $F_\beta score$. Esto se podía preveer fácilmente viendo que hay poca distinción en la gráfica de los resultados de clasificación. No obstante, el objetivo era comprobar hasta dónde puede llegar el modelo. Se puede ver en la siguiente figura que la curva pierde mucha pendiente antes de llegar a los valores cercanos al umbral utilizado.

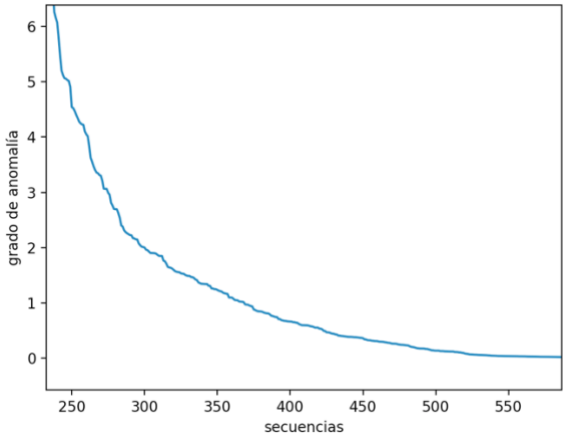

Figura 5-1: Gráfica de grados de anomalía

4. En esta prueba comienza la búsqueda del umbral perfecto mientras, paralelamente, se hacen pruebas cambiando parámetros del modelo. Se aumenta el umbral, pero se siguen encontrando 13 falsos positivos.
5. Se aumenta el umbral, pero siguen habiendo 3 falsos positivos. El porcentaje de acierto es el mismo que el de la prueba anterior, sin embargo el $F_\beta score$ mejora por el aumento de la precisión.
6. Se aumenta bastante el umbral aunque incluyendo pocas secuencias, ya que estas comienzan a tener grados más altos de anomalía. Es interesante ver cómo el acierto disminuye pero se obtiene el mejor $F_\beta score$ hasta ahora, valor por el que se evalúan los modelos en el reto.
7. En esta prueba se entrega exactamente la misma solución que en la prueba anterior, excepto por dos secuencias. Se analizan visualmente las secuencias que no se clasifican como anómalas en la prueba 6 pero sí en la prueba 5 para intentar identificar verdaderos positivos. Se identifican dos individuos con rasgos

característicos de las secuencias anómalas y se realiza la validación exitosamente al ver que ambos son realmente anómalos. Esto motiva la realización de la prueba 9 con el fin de identificar algunas de las secuencias del conjunto de falsos negativos.

8. Se selecciona el umbral con la precisión perfecta y el mejor recall hasta ahora. Esto proporciona un conjunto de datos anómalos perfecto para poder entrenar los modelos siguientes basados en entrenamiento supervisado.
9. Se realiza una prueba del modelo basado en el cálculo del coeficiente de correlación entre cada uno de los individuos del conjunto de datos de validación y el vector medio de los vectores de medias de todos los individuos normales. A pesar de que la idea de identificar los patrones con más tendencia de las secuencias normales parecía buena, el modelo muestra resultados que lo descartan por completo.

El modelo de k-nn se lleva a su límite y se optimiza hasta obtener, como mejor resultado, un $F_\beta score$ de 0.9699 lo que posiciona el proyecto en un buen lugar en el ranking de soluciones presentadas al reto de Airbus. Por otro lado, este modelo consigue detectar 205 individuos anómalos en el conjunto de datos a clasificar, lo que proporciona un potencial conjunto de entrenamiento para modelos de aprendizaje supervisado.

5.2 Resultados de pruebas con redes neuronales convolucionales

El modelo de redes neuronales convolucionales se elige por su constante uso en la comunidad de la ciencia de datos en problemas de clasificación de imágenes y por sus buenos resultados con los conjuntos de datos MNIST, CIFAR-10 y CIFAR-100 []. Se desarrolla el modelo basándose en ejemplos de otros modelos que han demostrado los mejores resultados en los conjuntos de datos mencionados anteriormente, los cuales tienen ciertas semejanzas al problema actual. Sobre este modelo se realizan algunos cambios en sus parámetros para conseguir determinar la arquitectura de la red óptima para el problema y entender qué elementos son cruciales para el obtener un buen resultado. A continuación se muestra una tabla con el conjunto de pruebas.

	$F_\beta score$	P	Recall	acierto	épocas	VP	FP	VN	FN
1	0.9811	1	0.8114	90.6%	30	241	0	297	56
2	0.9789	0.9854	0.9124	94.9%	10	271	4	293	26
3	0.8945	0.891	0.936	84.1%	30	278	34	248	34
4	0.986	1	0.8552	92.7%	30	254	0	297	43
5	0.988	1	0.8754	93.7%	30	260	0	297	37

Tabla 5-2:Tabla de resultados con cnn

Las pruebas realizadas se detallan en los siguientes puntos.

1. La primera prueba se entrena con los 205 individuos anómalos duplicados varias veces junto con los ejemplos normales del conjunto de entrenamiento. Este primer modelo se entrena durante 30 épocas, lo cuál se considera suficiente al ver que se obtiene un acierto del 100% en la validación. Consigue detectar 241 verdaderos positivos prometiendo buenos resultados con posibles ajustes por su precisión perfecta.
2. Se considera que es posible la existencia de sobreentrenamiento. En la época 10 de esta prueba se consigue un acierto en entrenamiento de 0.959 y en validación, de 0.988. Se detiene el entrenamiento en este punto sin esperar a obtener mejor acierto por las altas probabilidades de que los fallos de aciertos se deban a falsos positivos en el conjunto de datos sobre los que se entrena. En la prueba 1 se ve que el modelo obtiene precisión perfecta y, por otro lado, clasifica secuencias supuestamente normales como anómalas, lo que sostiene el argumento anterior. El modelo clasifica 275 secuencias como anómalas y obtiene 3 falsos positivos. A pesar de que el resultado se puede considerar mejor por un aumento en el acierto, la precisión baja ligeramente, lo cual no es deseable.
3. Se vuelve a entrenar con 30 épocas, pero esta vez cambiando la arquitectura de la red. Se mantienen los dos bloques pero se intercambian los números de filtros. El primer bloque pasa a tener 64 filtros y el segundo 32. Se realiza este cambio por la tendencia en muchos modelos populares en el estado del arte a disminuir el número de filtros en las capas más profuntas. Sin embargo el modelo no obtiene buenos resultados y disminuye el $F_\beta score$ en 0.08.
4. Se incluyen las anomalías detectadas en la prueba 1 en el conjunto de entrenamiento con el objetivo de que la red aprenda mejor los patrones que diferencian los individuos de cada una de las clases. Este cambio, con la misma arquitectura que en la primera prueba, muestra una mejora importante en los resultados detectando 11 secuencias anómalas más y obteniendo el mejor $F_\beta score$ hasta ahora.
5. Se repite el mismo proceso que en la prueba anterior. El conjunto de entrenamiento contiene 254 secuencias anómalas y el modelo es capaz de entrenar clasificándolas todas correctamente e incluyendo 6 verdaderos positivos más que antes.

El modelo con redes neuronales convolucionales confirma su buen comportamiento en el tratamiento de imágenes. Las pruebas realizadas concluyen que se consiguen resultados muy sólidos en precisión y demuestran que estas soluciones encajan perfectamente con el objetivo del concurso, el cual es desarrollar modelos que den favoritismo a la precisión sobre el acierto global.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

El tema de este proyecto ha sido la participación en un reto de aprendizaje automático propuesto por una empresa que ya aplica la inteligencia artificial en sus productos y está en busca de extender su uso en la monitorización de sus aeronaves. Además de simplemente participar en el concurso, se deseaba aprender a trabajar en el tema de la inteligencia artificial y, en especial con series temporales. Estos objetivos se han completado satisfactoriamente con el trabajo realizado a lo largo del proyecto. Se han realizado estudios de los datos y he aprendido a analizarlos para conseguir sacar la mayor información posible de ellos. Se han construido modelos de distintos tipos, con lo cual he adquirido el conocimiento necesario para, no solo aplicarlos, sino conocer sus diferencias y sus ventajas.

Se ha iniciado una primera aproximación para resolver el problema basada en el algoritmo k-nn para medir los grados de anomalía de cada secuencia. Se consiguió detectar suficientes anomalías como para tener un conjunto de datos de suficientes individuos de cada clase para usarlos en un algoritmo de aprendizaje automático supervisado. El modelo finalmente usado como mejor solución del problema planteado, se basa en el uso de redes convolucionales, las cuales representan la base del modelado de imágenes hoy en día. Esta solución final ha tenido resultados muy cercanos a los de los participantes que han liderado el concurso, dejando una buena posición final en el ranking.

Haber desarrollado una buena solución para un problema planteado en un concurso de inteligencia artificial con participantes de todo el mundo me ha familiarizado con los pasos y el proceso a seguir en este tipo de retos. Haber aprendido a participar y a desarrollar soluciones con la ayuda de Doroteo en este tipo de entornos, me abre la posibilidad de continuar participando en futuros programas similares. Con la idea de seguir mejorando y adquiriendo habilidades que me ayuden en el ámbito profesional, continuaré participando en el reto actual y consideraré trabajar en problemas similares a los que se enfrente el grupo de investigación AUDIAS, el cual trabaja constantemente en el tema de la inteligencia artificial aplicada a series temporales en problemas como el procesado de señales o de voz.

6.2 Trabajo futuro

Como trabajo futuro más inmediato, conviene revisar técnicas más sofisticadas de modelado de imágenes que se basen en los modelos usados en el proyecto. El modelo desarrollado con mejores resultados ha sido el de redes neuronales convolucionales. Este modelo es el más popular en el tratamiento de imágenes, pero como se ha mencionado en el estado del arte, esto es la base de otras redes más complejas que han demostrado tener ciertas ventajas con las convolucionales simples. El primer ejemplo consiste en las redes neuronales residuales (ResNet) que hacen un mejor uso del *deep learning* y solucionan problemas que las redes convolucionales simples sufren. Por otro lado, las redes *Wide ResNet* representan los últimos avances en el modelado de imágenes que se describen en el apartado del estado del arte. Realizar experimentos con estos dos tipos de redes neuronales aplicándolas a los espectrogramas puede concluir en mejoras de los resultados obtenidos con las convolucionales típicas.

Por otro lado, en el área del modelado de series temporales existen técnicas que aprovechan la información del contexto de cada individuo. Las redes LSTM, un tipo especial de las redes neuronales recurrentes, son usadas para este tipo de problemas ya que son capaces de almacenar información de los datos por determinados periodos de tiempo. Una aproximación

al reto al que nos enfrentamos en el proyecto con redes de estas características podría obtener buenos resultados y por ello, sería un buen trabajo futuro.

Por útltmo, los retos de inteligencia artificial aplicada a resolver determinados problemas que proponen empresas son una buena práctica para aprender y para conseguir una buena posición en la comunidad. Continuar participando en proyectos del mismo estilo también se puede considerar como trabajo futuro en lo que se refiere a mejorar y adquirir habilidades que nos ayuden a dominar este tipo de retos.

7 Referencias

[1] R. Nau, «Introduction to ARIMA: nonseasonal models,» 1 6 2018. [En línea]. Available: https://people.duke.edu/~rnau/411arim.htm. [Último acceso: 10 4 2019].

[2] A. A. Markov, The theory of algorithms, vol. 42, Moscú: Acad, 1954, p. 375.

[3] A. Karpathy, «cs231n,» 1 1 2015. [En línea]. Available: http://cs231n.github.io/. [Último acceso: 15 3 2019].

[4] Y.-L. Boureau, J. Ponce y Y. LeCun, «A Theoretical Analysis of Feature Pooling in Visual Recognition,» 2010. [En línea]. Available: http://yann.lecun.com/exdb/publis/pdf/boureau-icml-10.pdf. [Último acceso: 10 3 2019].

[5] J. Olafenwa, «Review of Identity Mappings in Deep Residual Networks,» 25 4 2018. [En línea]. Available: https://medium.com/deepreview/review-of-identity-mappings-in-deep-residual-networks-ad6533452f33. [Último acceso: 22 2 2019].

[6] R. Kumar Srivastava, K. Greff y S. Jürgen, «Highway Networks,» 3 11 2015. [En línea]. Available: https://arxiv.org/abs/1505.00387. [Último acceso: 22 2 2019].

[7] S. Zagoruyko y N. Komodakis, «Wide Residual Networks,» 14 6 2017. [En línea]. Available: https://arxiv.org/pdf/1605.07146.pdf. [Último acceso: 22 2 2019].

[8] S.-H. Tsang, «Identity Mapping - Over 1000 Layers Reached (Image Classification),» 22 9 2018. [En línea]. Available: https://towardsdatascience.com/resnet-with-identity-mapping-over-1000-layers-reached-image-classification-bb50a42af03e. [Último acceso: 23 2 2019].

[9] A. Krizhevsky, «Convolutional Deep Belief Networks,» 10 8 2010. [En línea]. Available: https://www.cs.toronto.edu/~kriz/conv-cifar10-aug2010.pdf. [Último acceso: 23 2 2019].

[10] A. Graves, Supervised Sequence Labelling with Recurrent Neural Networks, vol. 385, Toronto: Springer, 2012, p. 140.

[11] A. Bonner, «Getting Started With Google Colab,» 1 1 2019. [En línea]. Available: https://towardsdatascience.com/getting-started-with-google-colab-f2fff97f594c. [Último acceso: 2 3 2019].

[12] L.-Y. Hu, M.-W. Huang y S.-W. Ke, «The distance function effect on k-nearest neighbor classification for medical datasets,» 9 8 2016. [En línea]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4978658/. [Último acceso: 3 3 2019].

[13] A. Krizhevsky, I. Sutskever y G. Hinton, «ImageNet Classification with Deep Convolutional Neural Networks,» 30 9 2012. [En línea]. Available: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. [Último acceso: 10 4 2019].

[14] T. Ghouzam, «Introduction to CNN Keras,» 18 7 2017. [En línea]. Available: https://www.kaggle.com/yassineghouzam/introduction-to-cnn-keras-0-997-top-6. [Último acceso: 10 4 2019].